PROJECT MANAGEMENT IN DISTRIBUTED COLLABORATIVE ENVIRONMENT: THE IECOLLAB CASE STUDY

by
João Alberto Arantes do Amaral

Bachelor of Science in Mechanical Engineering (1992)
Escola Politecnica da Universidade de Sao Paulo

Submitted to the Department of Ocean Engineering and the Department of Civil and Environmental Engineering in Partial Fulfillment of the Requirements for the Degrees of

Master of Science in Ocean System Management
and
Master of Science in Civil and Environmental Engineering

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
September 2000

Signature of Author .....
Department of Ocean Engineering
July 28, 2000

Certified by .....
Joyce Warmkessel
Senior Lecturer, Aeronautics and Astronautics
Thesis Supervisor

Certified by .....
Joseph Martin Sussman
Professor of Civil and Environmental Engineering
and Engineering Systems
Thesis Reader

Certified by .....
Henry Marcus
Professor of Marine Systems
Thesis Supervisor

Accepted by .....
Daniele Veneziano
Chairman, Departmental Committee on Graduate Studies

Accepted by .....
Professor Nicholas Patrikalakis
Kawasaki Professor of Engineering
Chairman, Departmental Committee on Graduate Studies

PROJECT MANAGEMENT IN DISTRIBUTED COLLABORATIVE ENVIRONMENT: THE IECOLLAB CASE STUDY

by

João Alberto Arantes do Amaral

Submitted to the Department of Ocean Engineering on
July 28, 2000 in partial fulfillment of the requirements for the
Degree of Master of Science in Ocean System Management
and Degree of Master of Science in Civil and Environmental Engineering

Abstract

Nowadays management distributed collaborative projects are becoming more and more complex. New tools are being used to make it possible to fully understand the nature of the project and the dynamic relationships involved. This research presents and reviews a significant number of principles, methods, models and computer tools used in project management in distributed collaborative environments.

Thesis Supervisor: Joyce Warmkessel

Title: Senior Lecturer, Department of Aeronautic and Astronautics

Thesis Reader: Joseph Martin Sussman

Title: Professor of Civil and Environmental Engineering and Engineering Systems

Thesis Supervisor: Henry Marcus

Title: Professor of Marine Systems

ACKNOWLEDGMENT

I want especially to thank Professor Joyce Warmkessel for her invaluable help, assistance and encouragement.

Professor Henry Marcus, for being such a helpful supervisor during these years at MIT.

Professor Joseph Sussman, for his assistance, good will and helpful comments.

Patricia Brennecke, for helping me to correct the grammar mistakes and for being such a good friend during these years at MIT.

Helena and Renato Pakter for their friendship and support.

Jean Sucharewicz and Beth Tuths for their assistance and good will.

I would also like to thank Professor Feniosky Peña-Mora and the thirty-four students from the ieCollab Project.

DEDICATION

I wish to dedicate this to my family, especially to my brother Eduardo. Thank you all for always being there for me.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Overview

Project management activities are crucial to the development of new products. The management of distributed collaborative projects involves dealing with delays, cost overruns, scheduling slippage, costly reworking, overtime and other problems (Sterman, 1992). The consequences of inefficient project management can be disastrous to a company's performance. Poor project management can result in poor profitability, loss of market share, poor product quality and many other problems. In order to avoid these problems, companies are using more and more computer models and tools to improve the management of distributed collaborative projects. The aim of this thesis is to cover the basic issues involved in project management in distributed collaborative environments.

1.2 Objectives

The thesis has three objectives:

1. To provide an overview of the basic concepts of project management applied to distributed collaborative environments.
2. To analyze the ieCollab project management activities as an example.
3. To contribute to further research in the area.

1.3 Methodology

The research methodology adopted focused on issues related to project management and the development of management skills in order to become able to manage distributed collaborative projects. The following activities were developed:

- Extensive research of System Thinking, System Dynamics and Quality Assurance models and tools, provided by courses at the Sloan School of Management.
- Review of basic concepts and models used in Project Management, provided by courses at Civil and Environmental Department and the Da Vinci Research Group's meetings and activities.
- Familiarization with project management and collaborative software tools, primarily the software Primavera, Vensim, NetMeeting, ICQ.
- Familiarization with real project management problems. This was achieved by working for two consecutive terms as a member of the project management team of the ieCollab research project.

1.4 Thesis Organization

This thesis is composed of nine chapters. Chapter 1 describes the thesis structure and chapter organization. It also describes the ieCollab project, the project referred in all following chapters. Chapters 2 presents an overview of the basic concepts and definitions involved in the management of distributed collaborative projects.

Chapter 3 presents the model that will be applied. Its basic components are project preparation, project planning, project execution and project adaptation.

Chapters 4 through 7 explain each one of the aforementioned components. The focus of this thesis is not to review deeply all activities involved and all methods used in project management. There are several books that cover this in detail (Rosenau, 1998), (Stubb et. al.,1994), (Roman, 1986), (Meredith et. al. 1989), (Krezner, 1989). Our idea is to give an overview of the main issues involved in each activity and show how they are applied to a real project, the ieCollab project.

Chapter 8 discusses the lessons learned in ieCollab project and Chapter 9 presents our conclusions.

1.5 *The ieCollab Project*

The ieCollab (**i**ntelligent **e**lectronic **collab**oration tools) was an Internet-based collaborative application software project that was conducted by students from MIT-USA, PUC-Chile and CICESE-Mexico. It was a short project that began in November 1999 and ended in April 2000. The team was composed of 34 students geographically dispersed: five from PUC, Chile; five from CICESE, Mexico; and 24 from MIT. I had the opportunity to work as project manager on the ieCollab Project Management team; my experience and analysis of this will be used throughout this thesis as the basis for observation on distributed, collaborative projects.

The scope of the project was to develop a software tool that could allow communication among remotely-located teams, providing meeting management facilities for other Websites (like Yahoo, Lycos etc), allowing documents and software sharing (models, simulations, calculations, drawings) and charging for the use of these facilities (Figure 1.1).

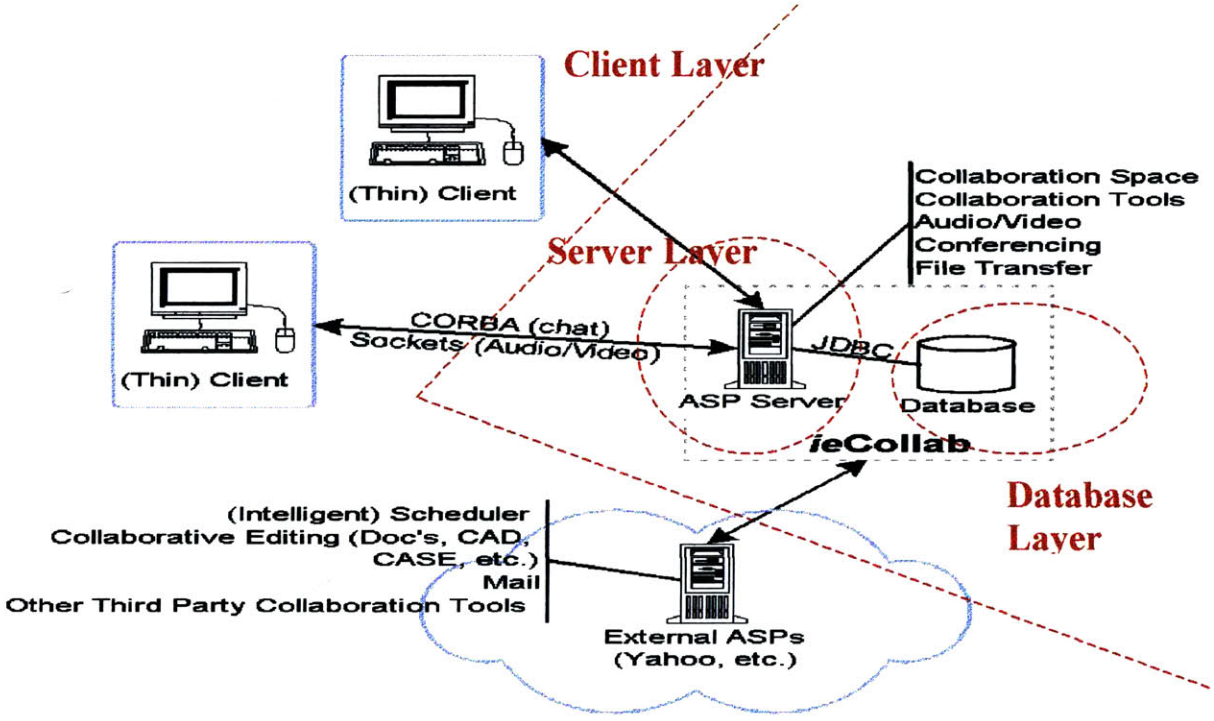

Figure 1.1 The ieCollab Project (Ref. Limansky et. al., 1999)

The ieCollab system uses the three-tiered client server architecture (Connoly et. al., 2000)

1. User interface layer (Client layer)
2. Business logic and data processing layer (Server Layer)
3. Database management system (Database Layer).

For further information and technical details about the ieCollab project, consult the project Website http://www.collaborate.mit.edu/FTP/P1. In this thesis we will not cover the technical aspects of this project; instead we will focus on the project management activities.

1.6 Summary

This chapter presented an overview of the thesis organization and overview of ieCollab project. In the next chapter we will begin with the definitions of the basic issues related to project management in a distributed collaborative environment. In order to understand what distributed collaborative project management means we will discuss the basic concepts of models, systems, project, project management and distributed collaboration.

CHAPTER 2

BASIC DEFINITIONS

In this chapter we will discuss the basic concepts of project management. We will begin with the definition of systems, models and project. We will cover project characteristics, constraints, elements and project lifecycle. Finally we will define project management and will study project management characteristics. We will end this chapter by discussing the main issues of project management in a distributed collaborative environment.

2.1 Systems

A system can be understood as a set of interdependent parts or components that interact with each other in order to achieve a final purpose. There are many different kinds of systems-- mechanical systems, electronic systems, economic systems, production systems, and so on. Companies are also systems; however, they represent another form of system called a social system. Jay Forrester (Forrester, 1971) explains that a "*social system belongs to the class of called multi-loop non-linear feedback system*".

Nowadays many companies are using models in order to represent their productive processes and to understand fully the consequences of decisions taken.

2.2 Models

Models are representations of parts of the systems or of the system itself. Models can be used to represent anything from a small and simple mechanical component to a complex industrial system. In general, we can say that models are created to simulate the behavior of the entity. We create models in order to have a better understanding of how the real system will behave when submitted to a set of conditions.

"*Model is a substitute for some real equipment or system. The value of a model arises from its improving our understanding of obscure behavior characteristics more effectively than could be done by observing the real system*"- (Forrester, 1961).

There are many different model classifications; however we will focus only on two generic types of models: mental models and formal models. A mental model is our idea of how a system works. It is not explicit; it has no formalism. Indeed, to explain the same process, different people will make use of different mental models. They are generic, flexible and broad. They have ambiguities and they are not clear.
Sterman (Sterman, 1998) teaches that:

"*Mental models stress the implicit casual maps of system we hold, our beliefs about the network of causes and effects that describe how a system operates, along with the boundary of the model and the time horizon we consider relevant-our framing or articulation of a problem.*"

Formal models are those used to make our mental models explicit. This can be done by using mathematical equations or any other tools to describe our ideas, our mental assumptions. Formal models make our implicit assumptions explicit. They can be used

to demonstrate the validity of an assumption or to make predictions, for example. When a formal model is created to describe a specific problem, people who have different mental models for that problem can reach an agreement, a consensus. Formal models are used to replicate the system behavior. However, most models are valid within particular limits. Outside of these limits the model can fail or be inaccurate. Furthermore, most systems have peculiarities that sometimes cannot be captured by any model. Models represent only a part of a real system. Models help to understand the system, but they are not a perfect representation of a system. Sterman (Sterman, 1992) points out the following characteristics of mental models and formal models (table 2.1).

	Mental Model	**Formal Model**
Advantages	▪ Flexible ▪ Take wide range of information into account ▪ Process information in variety of forms ▪ Easily adapted to the new situation and modified	▪ Explicit ▪ Assumptions are open to all for review ▪ Able to interrelate many factor simultaneously ▪ Controlled conditions
Disadvantages	▪ Not explicit ▪ Not easily examined by others ▪ Interpretations differ ▪ Has ambiguities and contradictions	▪ Can be misused

Table 2.1 Characteristics of Mental Models and Formal Models (Sterman, 1992)

Someone once said that all models are wrong, but some are useful. That is the key idea. A model is good, if, in spite of its limitations, it can give us useful insights. A good model is the one that can bring to the surface people's mental models (Senge, 1990).

Nowadays models are used by companies in the learning process. Models are used not only to make simple the understanding of the dynamic relationships involved but to represent the project itself.

2.3 Project

In order to understand what project management is, first we need to understand what a project is.

" *A project may be viewed as the entire process required to produce a new product, new plant, new system, or other specified results*"(Roman, 1986)

A project coordinates a sequence of activities necessary to create a new product. It involves the management of people and resources. It requires planning and scheduling. It needs continuous control and monitoring. The project's size and duration depends on the complexity of the product to be developed.

2.3.1 Project Characteristics

Projects, in general, have the following characteristics (Thomsett, 1990);(Rosenau, 1998):

▪ **Exception**	A project is not a routine; it is outside the range of usual routine activities. Every project is different, even if the final products are quite similar.
▪ **Related activities**	The project involves coordinate teamwork; one activity depends on previous activities.
▪ **Goals and deadlines specifics**	Projects have goals, a start point and a completion date. In order to achieve the goals, milestones are established and deadlines are specified.
▪ **Desired results**	All projects aim to a final result. The final result of a project can be a new product, a report, a technology demonstrator, new software, and so on.
▪ **Involves uncertainties**	Projects involve risk and uncertainties in different degrees.
▪ **Temporary**	The project is finite; there is a specific date to be finished.
▪ **Resources**	People, material and money are resources used in projects.
▪ **Organization**	Projects are generated by organizations. Organizations are dynamic and complex, which can influence the way that the project is done.

Table 2.2 Characteristics of projects (Thomsett, 1990); (Rosenau, 1998)

2.3.2 Project Constraints (Triple constraint)

There are basically three constraints to the project (Rosenau, 1998):

- Results (Performance specification)
- Budget (Money)
- Time

The project should be finished at the date planned, with the technological level desired and within the planned budget. However, many projects face problems such as schedule slippage and cost overrun. Indeed, many end without achieving the desired performance level. Some researchers (Peña-Mora *et. al.,* 1999) point out that the average project overrun (time, money or both) is between 40% and 200 %.

2.3.3 Project Elements

Some authors (Roman, 1986) define the elements of the project as follow:

- Task: what is needed to be done
- Methods: how to accomplish the goals
- Resources: all that is needed to accomplish the goals (personnel and equipment).
- Environment: where the project is done, the stakeholders involved

2.3.4 Project Life Cycle

The project's life cycle is associated with the kind of product that we intend to build. The lifecycle of a software project is different from that of a ship, for example, but both share basic principles. Basically, in any project the initial step is to define the requirements or specifications. The second step is the analysis, to define all alternatives possible feasible to implement and choose one. The third step the design and the last step is the fabrication, to make the project, to build the system. The last step is testing in order to check if the system does what it was specified to do. There are many different types of project life cycle depending on the specific nature of the project.

In general the project resources increase continuously from the beginning, reach a maximum at the implementation phase and decrease continuously at the end (figure 2.1).

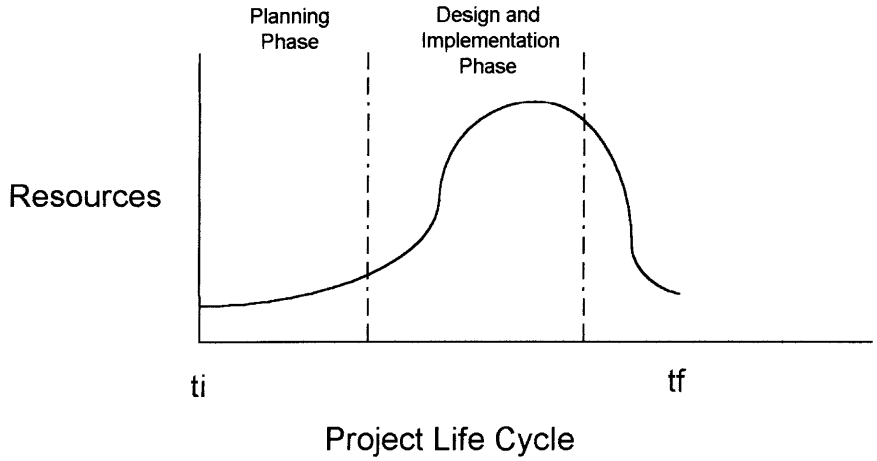

Figure 2.1 Project Life Cycle (Krezner, 1989)

2.4 Project Management

There are several definitions of project management. Let us take two as examples:

"*Project management, in large part, is the management of interpersonal conflicts, which is inherent in complex organizational situations*" (Rosenau, 1998).

"*Project management is a departure from management in cycled activities where risk and accomplishment can, with some degree of probability, be anticipated*" (Roman, 1986).

In general the definitions of project management will stress the importance of interpersonal conflicts and risk. A similar project with similar resources and characteristics can have a totally different time duration and resource consumption (Peña-Mora *et. al.*, 1999). The way of managing the conflicts, reworking and mitigating risks makes the difference (Figure 2.2).

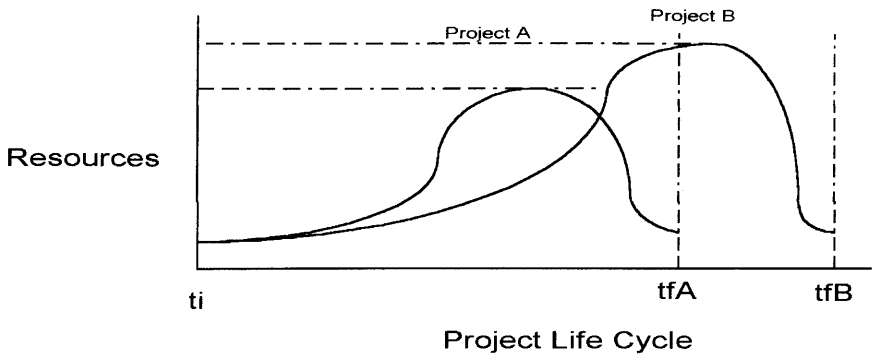

Figure 2.2 Differences between similar projects (Peña-Mora *et. al.*, 1999)

2.4.1 Project Management Characteristics

Management of large scale projects, in general, have the following characteristics (Sterman, 1992):

Characteristics of large scale projects	**Brief description**
Complexity	▪ Multiple interdependent components ▪ Change in one part will have implications in others
Highly dynamic	▪ System's short term response can differ significantly from the long term response
Multiple feedback process	▪ Presence of reinforcing and balancing loops
Non-linear relationship	▪ Causes and effects do not have proportional relationship
Involve "hard" and "soft" data	▪ Involves engineering issues ("hard") and managerial decisions ("soft")

Table 2.3 Management of large scale projects (Sterman, 1992)

2.5 Project Management in a Distributed Collaborative Environment

The advent of the Internet is changing the traditional way that corporations organize themselves to handle projects. There is a clear trend toward the change from project-based work to team-based work (Grantham, 2000). The project's work activities are accomplished in parallel by globally dispersed teams. One clear example of this new paradigm is the Boeing 777 project. The project involved 7,000 engineers, divided in 240 design-build teams (Smith et all, 1998). The project involved teams located in more than dozen countries, connected by a network of 1,700 workstations (Benson-Armer, 1997).

Townsend defines a virtual team as a "*group of people, geographically and/or dispersed coworkers that are assembled using a combination of telecommunications and information of technologies to accomplish an organizational task*" (Townsend et. al., 1998).

The ieCollab project is an example of a virtual team. The team was composed of 34 students from three different countries (USA, Mexico, Chile) and three different organizations (MIT, CICESE, PUC) working on a unique project.

Horvath (Horvath et. al., 1999) stresses the main characteristics of virtual teams:

1. The teams are not restricted by boundaries
2. The communication is done through information technology (Internet, Intranet, Shared Space such as Website repository, and email)
3. The membership is dynamic and adaptable

There is an important issue related to the communication via information technology. Smith (Smith et. al., 1998) stresses:

"*Studies in human communication reveal that less than half of what is conveyed in a conversation comes from words themselves; the majority of the message is communicated via intonation, body language and timing. Electronic communication, depending on the medium involved, often fails to capture those other elements.*"

Indeed, if we analyze the most common electronic means used in virtual team communications we will see that all of them have serious problems. We will discuss this issue in details in Chapter 4.

2.6 Summary

Our goal in this chapter was to give an overview of models and project management for those who are not familiar with these topics. In the next chapter we will discuss the model that we are going to follow in order to study the management of distributed collaborative projects.

CHAPTER 3

Framework for Managing Projects

In this chapter we will present the general framework of the project management model that will be used as a basis for analyzing the ieCollab Project.

3.1 The Project Management Model

Figure 3.1 shows the steps involved in the management of a project.

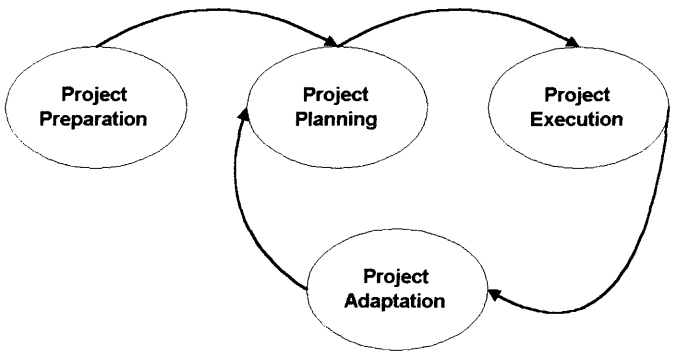

Figure 3.1 Project Management Model

Each one of the previous steps presented is composed of a set of activities that are represented in figure 3.2.

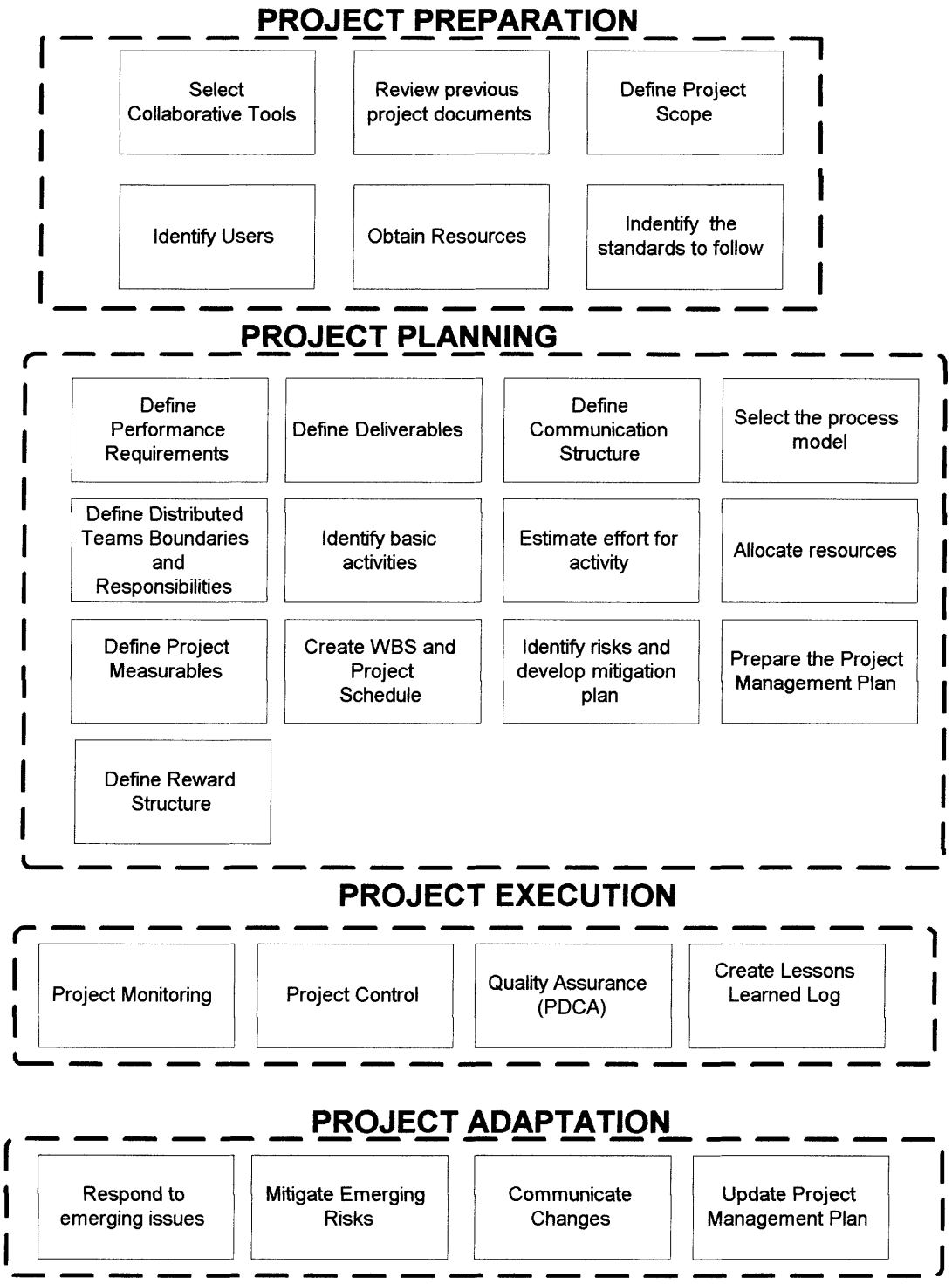

Figure 3.2 Activities in Project Management Model

3.2 Project Preparation

The project preparation begins with the definition of the project's scope and the identification of the end users. In this phase resources are obtained and standards to follow are defined. Previous projects are analyzed when they are available. In a distributed collaborative environment, the teams are formed and collaborative tools are selected. We will cover project preparation in Chapter 4.

3.3 Project Planning

Project planning begins with the definition of the deliverables, the determination of performance requirements and the selection of the process model. The communication structure, the team's responsibilities and the reward structure are also settled.

After the previous issues are defined, the next step is to establish of the work breakdown structure (WBS) and to create the project schedule. The WBS shows the basic components of the project, the work packages. A work package can be composed of one or several activities. Each activity will have its duration based on the amount of resources allocated. Once the activity's duration has been defined, the project schedule can be created. The schedule defines the network of the logical sequence of tasks to do.

In order to determine if the project will unfold the way it was planned, it is necessary to define how to measure project progress.

Another important issue of the project planning is risk identification and mitigation.

We will cover the basic planning concepts in Chapter 5. In order to make clear the concepts involved in planning, the ieCollab project management plan will be analyzed in detail.

3.4 Project Execution

The project execution has four major components: project monitoring, project control, quality assurance and creation of lessons learned database. The monitoring process is used to make sure that the project is following the plan.

The monitoring process also involves verifying that the product in development performs the functions that it is supposed to and has the desired quality. Basically the project manager should have mechanisms that help him/her monitor the activities, check if they are following the planned schedule and verify if the costs are between the budget limits.

Project control refers to the actions taken in order to correct the deviation from the plan revealed during the monitoring activities. The most common actions are changes in resources and personnel re-allocation, changes in budget and alterations of scheduled activities.

It is also important that the organization learn lessons from a project. Lessons learned in previous projects are useful data for new projects. The main interest of post-project learning activities is to summarize and disseminate the record of the lessons learned in the project that has been completed. The record of lessons learned helps to create a mitigation plan for future projects. Lessons learned in previous project enable the new team to avoid previous mistakes. The study of previous projects can also provide data that can be used in models for the new project. We will discuss the project execution issues of the ieCollab project in Chapter 6 and the lessons learned in Chapter 8.

3.5 Project Adaptation

Planning, monitoring and control are intermingled activities. Monitoring attempts to determine the planning is being followed, to see the deviations. Based on the monitoring information, the project manager can take corrective control measures. These measures will be taken in order to respond to emerging issues and to mitigate emerging

risks. During this process the project management plan will be updated and the changes will be communicated. These are the main components of project adaptation.

The project management plan is not a static document; it changes over time due to changes in assumptions, emerging risk, delays and other problems (Krezner, 1989). The initial product assumptions can change when the developer's team has a more detailed and clear idea of the product itself. Some activities not anticipated can become necessary to accomplish and new tasks should be added to the main schedule. The budget can also change during the project's life; tighter budgets can imply on changes in the amount of raw materials and equipment that were planned to be used (Krezner, 1989). Technology also change along of the product development process. In order to avoid the products becoming obsolete soon after its release, the developer's team should be willing to incorporate current technology into the new product. The developer's team size also changes over time. People leave the project, new people come in to the project, training activities may be scheduled, new activities may be added to the main schedule, and the length of activities may be modified.

We will cover the main issues of project adaptation in Chapter 7, when we discuss how we responded to emerging issues and mitigated risks in the ieCollab project.

3.6 Summary

In this chapter we gave an overview of the activities of the project management model. In the following chapters we will describe each model component with emphasis on issues that apply to distributed collaborative projects, using the ieCollab as the basis for analysis. We will begin with project preparation, the topic of Chapter 4.

CHAPTER 4

Project Preparation

During the project preparation phase, the project management team will define the project's scope and will obtain the necessary resources and tools. In this chapter we will focus on issues related to the preparation of a collaborative distributed project. We will discuss preparation for the ieCollab project as an example.

4.1 Selecting Collaborative Tools

All projects rely on teamwork and collaboration. Distributed collaborative projects have the added complexity that the teams are not co-located. In the project preparation phase, one of the most important issues is to establish an effective communication channel among the different teams. Collaborative tools make this communication process feasible.

As we discussed in Chapter 2, the most common electronic means used nowadays by virtual team communications are E-mail, Website repositories, Video-Conferencing and Web-conferencing. All of them have serious problems, summarized in table 4.1 (Smith et. al., 1998).

	Advantages	Disadvantages
E-mail	▪ Effective for non-urgent communication	▪ Ineffective when rapid response is necessary.
Website repository	▪ Fast way to transmit and share documents	▪ Difficult to explain the nuances of the documents
Video-Conference	▪ Allows face to face meeting	▪ Fails to capture body language and timing ▪ Expensive
Web-Conference	▪ Allows face to face meeting and document sharing	▪ The technology is still immature. Connection problems, poor image definition

Table 4.1 Advantages and Disadvantages of Collaborative Tools (Smith et. al., 1998)

Collaborative tools help to provide the necessary communication channel, but they are only part of solution. The most difficult issue is to assure that those tools are being used effectively. It is necessary to build solid trust relationships among the teams at the beginning of any collaborative project. Otherwise the tools will not be used to their full potential. There is a general consensus nowadays that it is important to provide an initial face-to-face meeting at the beginning of the project.

There are many other issues related to the distributed collaborative work such as cultural differences, time delays and lack of trust (Benson-Armer, 1997). As ieCollab was an educational problem, it was not possible to have face-to-face meetings between MIT students and CICECE and PUC peers at the beginning of the project. We intended to meet the remote teams at the end of the project preparation phase, but problems prevented this from happening. However, steps were taken to mitigate this issue.

All students had personal email and access to the project's Website Repository (Active Project[PM] Server) .We used Web-Conference software (NetMeeting and ICQ) in order to establish contact with students in remote locations. We talked each other during the initial project preparations. In order to know each other better, the students were asked to make personal homepages for the remotely located students.

The Web Site repository (Figure 4.1) was used from the beginning of the project. In it we were able to store all project documents and make comments about their contents.

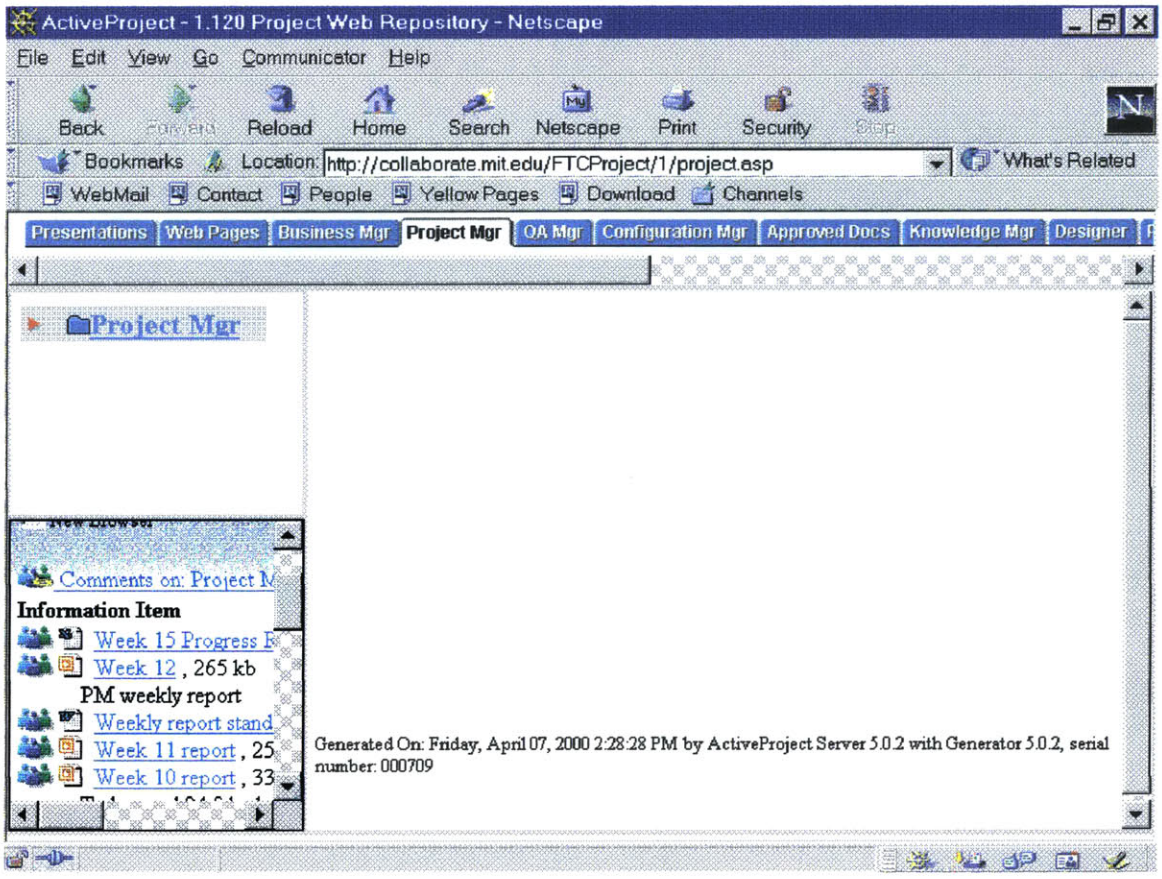

Figure 4.1 Web Site Repository (Active Project[PM] Server)

We also made use of a Web-repository for approved documents and software code. This repository was maintained by the Configuration Management team (CM) and by the Knowledge Management team (KM). The KM team was responsible for the approved documents. The CM team was responsible for the code control. This Web-site had a link to the CVS repository, a software tool designed for control of software code.

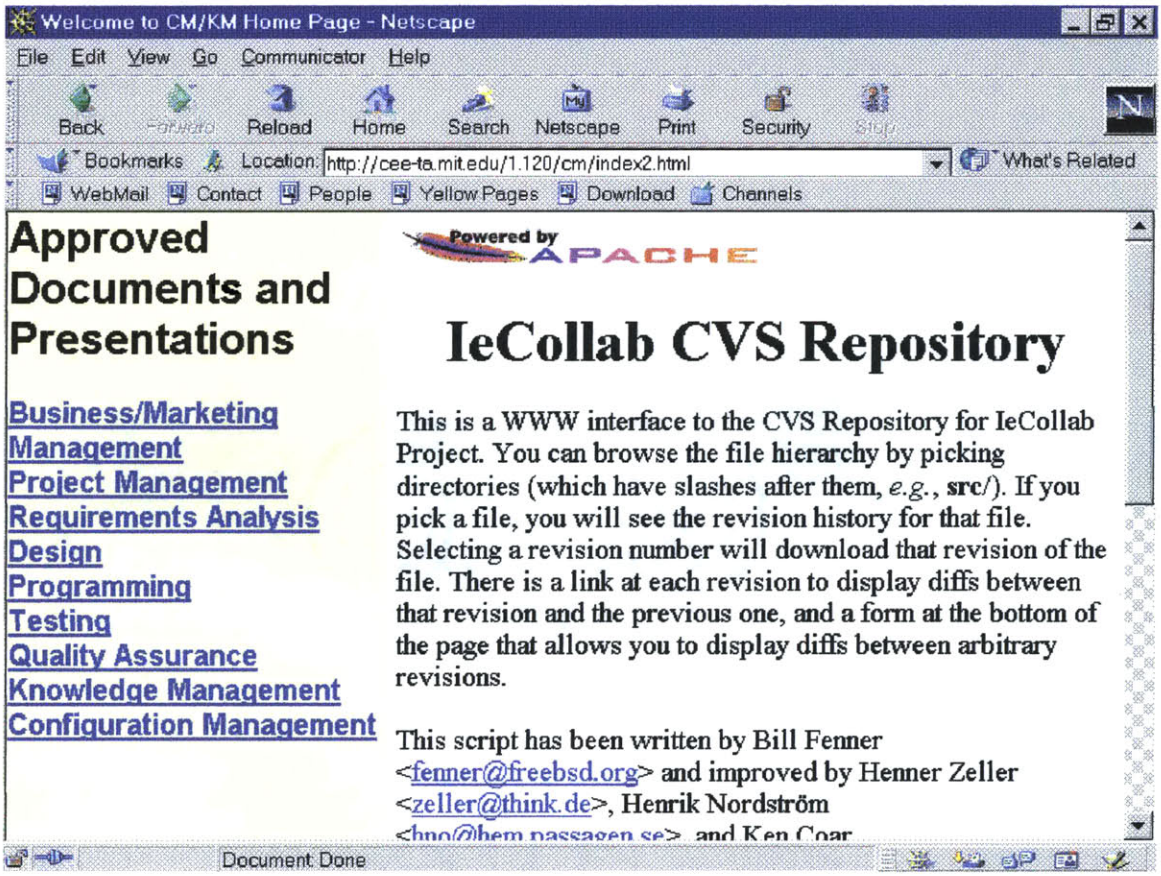

Figure 4.2 The ieCollab CVS Repository

In spite of all these resources and the initial measures taken, when the project unfolded the communication among teams began to degenerate. We will discuss the communication problems in Chapter 8.

4.2 Review of previous project's documents

The review of previous projects is fundamental in helping to avoid repeating the same mistakes. The data of previous projects can also be re-used. Data reutilization helps to speed up new projects. The previous project's documentation can be used as the start point for new documents.

In software projects, code re-utilization is gaining more and more attention. However, the ieCollab project was not very successful in reusing material from the previous year's project, the Cairo project. The standards used in the Cairo Project and the planning documents help us a guideline, a start point for our planning. In this aspect it was very useful. However, the re-utilization of technical documents was not successful. The analysis, design and code were not re-used at all. This may have been because the software developed in the previous year had many code errors. We tried to fix the Cairo errors initially, but without success. The teams soon realized that it would take more time to fix the Cairo code than to build the ieCollab code from scratch.

4.3 Defining the project's scope

The definition of scope describes the project's boundaries: what the product's features (functions and performance) will be, what the context constraints are, when delivery will be, when the project will be terminated, what the success metrics are and what the resources are. In ieCollab project we did not have a clear definition of the project's scope. We had some boundaries, and based on that we defined the project's scope.

The ieCollab project was defined as follows (Amaral et.al., 1999):

"The software package will be internet-based, allowing geographically independent collaboration. Meeting management facilities will be incorporated, and the software will double as a mode for application and document sharing.

Main Characteristics:

- *Internet-based collaborative application*

- *Document Sharing*
- *Application Sharing*
- *JAVA meeting environment (platform-independent)*
- *Meeting management facilities (provided by other web-sites like Yahoo, Lycos, etc.). "*

In our definition of project scope we followed Pressman's definition of the scope of software projects (Pressman, 1997), we covered the context, information objectives and functions. We did not cover the context constraints. As the project unfolded it became clear to us that we had to refine our definition of scope. Our initial project scope seemed to us impossible to achieve in the time that we had. We will cover these aspects in Chapter 9.

4.4 Identifying end users

In general, software projects are undertaken in order to create a product that will address a specific market segment. Usually software can be created to address general needs (such as office tools, CAD-CAM tools, collaborative tools) or can be created at the request of a specific customer. The main issue is to find out who will use the software and in what environment it will work (Pressman, 1997).

4.5 Selecting Standards to Follow

Humphrey (Humphrey, 1990) defines a standard as:

" A standard is a rule or basis for comparison that is used to asses size, content, value or quality of an object or activity.
...the fact that everyone knows and understands a common way of doing the same tasks makes it easier for the professional to move between projects, reduces the need for training and permits a uniform method for reviewing the work and its status. "

There are standards for the development process and standards for the product. In the ieCollab project, at the beginning of the project the quality assurance team together with the project management team made a selection of all standards that we were going to follow. All standards we choose were IEEE standards. Copies of the standards were provided to all teams, and the Quality Assurance team made sure that the standards were being followed throughout all the project's lifetime.

We used the following standards in order to create the our project management plan:

- IEEE Standard for Software Project Management Plan
- IEEE Standard for Software Productivity Metrics
- IEEE Standard for Developing Software Life Cycle Processes
- IEEE Standard for Software Quality Assurance Plan
- IEEE Standard for Software Quality Metric Methodology

4.6 Resources

Project resources are the personnel, material and equipment used in the project. The following resources were used in the ieCollab project:

Personnel	34 MIT/PUC/CICESE students
Hardware	Computers at MIT- Master of Engineering room, Design Studio, and Athena Clusters; CICESE and PUC laboratories.
Software	Development tools: Java 2.0, Corba, Rational Rose Database tools: Oracle Office tools: MS-Word, MS-Excel, MS-Powerpoint Project tools: MS-Project, Primavera Communication tools: Neetmeeting, ICQ Web-repositories: Active Project^{PM} Server, CVS

Table 4.2 ieCollab Resources

4.7 Summary

In this chapter we reviewed some issues of project preparation. Reviewing previous projects, selecting collaborative tools, identifying end users, selecting standards and obtaining resources are some of the most important activities in this phase of the project. In the next chapter, we will cover the main aspects of project planning, using the ieCollab project as a case study.

CHAPTER 5

Project Planning

In this chapter we will discuss the main issues involved in project planning. For each issue covered we will give an example of how we used it in the ieCollab project plan.

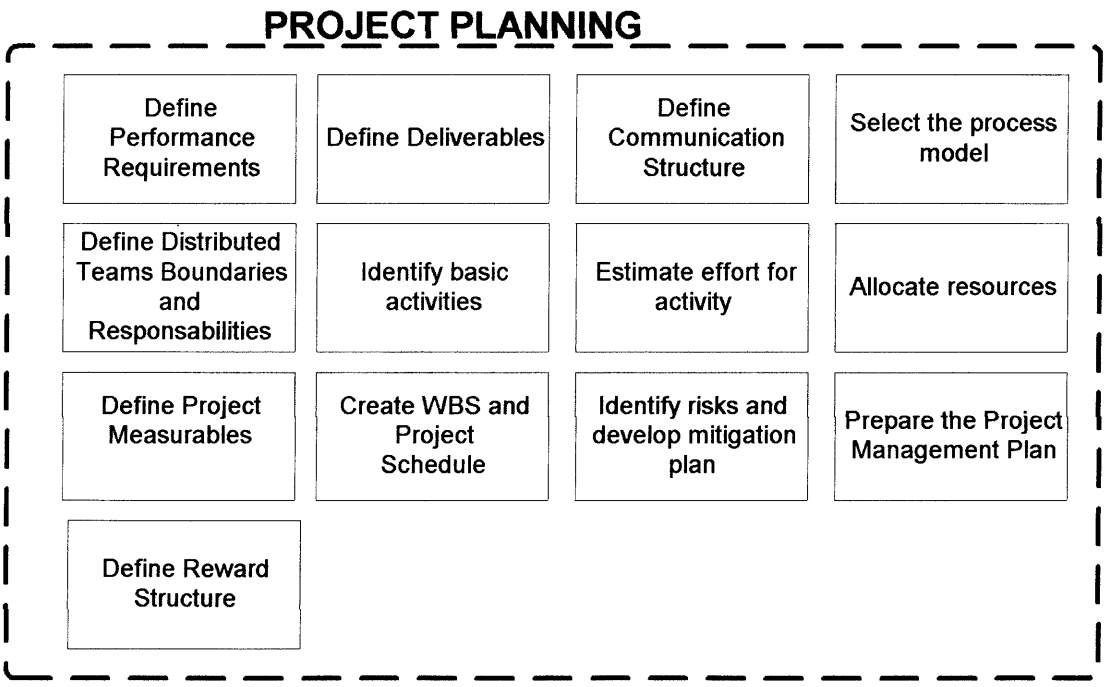

Figure 5.1 Project Planning Activities

5.1 Performance Requirements

Performance requirements are the desired features of the final product. In software projects, performance can be defined in terms of portability, speed, graphical interface, database characteristics, client-server architecture, multimedia resources, interfaces with other software package and internet-based characteristics.

In the planning process, only general goals for the product are stated. In general, the business managers and the market managers are responsible for seeing to this. The translation of the performance goals (defined by the market/business managers) into formal specifications is done during the execution phase by the requirement analysis (RA) team. The RA team will use various modeling tools (Use Cases, for example) to specify the desired functionality of the software package, indicating how it interfaces with other systems and the constraints involved.

5.2 Deliverables

Deliverables are the main components of the project. In software projects deliverables are not only the code, but the technical documents and user's guide as well. In the ieCollab project we defined the following deliverables:

- CD with software for Version 1 and 2, and User's Guide
- Hard Copy of User's Guide
- Technical Guide
- On-line site, with all documents, presentations and comments.

5.3 Communication Structure

Communication structure defines how the information will flow among the teams, which communication tools will be used, how often meetings will be held and how the decisions will be reported to all teams.

In the ieCollab project the communication among teams was achieved via e-mails and ICQ (Web-Conference software). Video conferencing and telephone calls were used sparingly due to the large costs involved. We defined the following report mechanisms:

- Weekly meetings of all teams with the project management team.
- All written reports were posted at project the Web-repository, and all documents followed the format specified by the Project Management Plan.
- Comments and discussions about the documents were also posted at the project's Web-repository, following a specific format .It was defined that the Quality Assurance team would be responsible for to making sure that all documents posted adhered to the standards. The leaders of each team were asked to read the comments posted in their respective sites and discuss them with the project management team on a weekly basis.
- Project walkthroughs/inspections lead by the Quality Assurance team
- Project manager announcements. These announcements were be made mainly during Thursday classes or by email.
- Team presentations.
- Emails, video conferencing, and telephone calls with virtual teams.

5.4 Distributed Team Boundaries and Responsibilities

The project manager of a distributed collaborative team is responsible for defining the work that will be performed by each team and the team's responsibilities. The project manager is also responsible for defining the leadership of the project. In the ieCollab project we decided to distribute the leadership of the project among the students of these three universities. Almost all teams had one component from a remote located university (Table 5.1). For example, the RA team had 03 team members from MIT, 01 from CICESE and 02 from PUC.

Teams	Number of Participants		
	MIT	CICESE	PUC
Project Managers	03	01	---
Analysts	03	01	02
Designers	02	01	02
Programmers	02	---	---
Testers	02	01	---
Quality Assurance	03	01	---
Knowledge Managers	02	---	---
Configuration Managers	02	01	---
Marketing Managers	02	---	---
Business Managers	02	---	01

Table 5.1 Team Structure

In the project management plan we defined the responsibilities of each team. The following paragraph shows how we defined the responsibility of the project management team (Amaral, 1999):

"Project Management Team:

Responsible for project organization, planning, monitoring and control. The project manager is responsible for creating the project management plan and taking necessary measures to assure that the plan will be followed. This team is also responsible for allocating resources to among the others teams where need be, giving necessary support, coordinating all activities, and maintaining efficient communication."

5.5 Activities Identification

Software projects involve development activities, project support and business activities (table 5.2). The development activities are the core of the software development process. Activities include such the technical tasks as requirement analysis, design,

coding and testing. The project support activities are those activities that organize and help the development process. The Project Management team defines how the project will be done; the Quality Assurance team will define the product and the process quality; the Configuration Management team will deal with management and control of code generated and the Knowledge Management will take care of all technical documents and data generated. Marketing Management team will define the market needs and will address the time to market issues.

Development Activities	▪ Analysis ▪ Design ▪ Code ▪ Test
Project Support Activities	▪ Project Management ▪ Quality Assurance ▪ Configuration Management ▪ Knowledge Management
Marketing Activities	▪ Business Management ▪ Marketing Management

Table 5.2 Project Activities

5.6 Selecting the process model

There several different software process models. The most commonly used are the linear sequential model, the prototyping model, the spiral model and the incremental model. In the ieCollab project we decided to use the incremental model, a model that combines characteristics of the linear and prototyping models (Pressman, 1997). The incremental model allows developing prototypes simultaneously. For example, we can have teams working in Analysis and Design of different prototypes simultaneously (Figure 5.2). This model was chosen because it allows the diverse teams to work in parallel, speeding up the development process. Two fully operational packages were

planned to be created in order to achieve the project goals defined by the Business Manager Team (Table 5.3).

Package 1 -Meeting Management Environment: allows distrubuted users to set up and manage online meetings (Li-Wei et. al.,1999)
Package 2-Transaction Management: allows the ieCollab server to track usage of ieCollab's meeting management service and charge fees on per-transaction basis (Li-Wei et. al.,1999)

Table 5.3 Operational Packages

From now on we will call[1] Package 1.0 and Package 2.0 of Version 1 and Version 2.

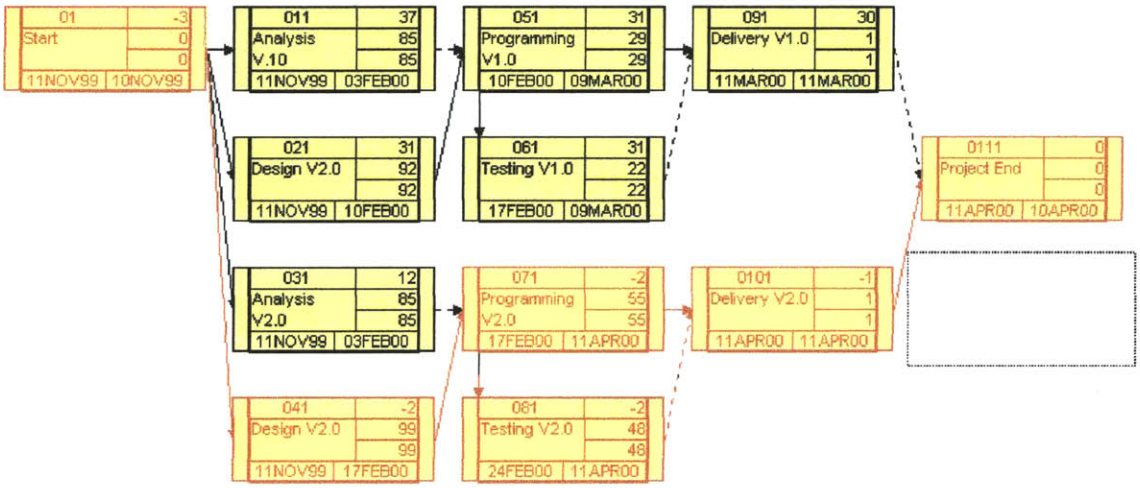

Figure 5.2 Project Schedule

In order to optimize our resources, we planned to develop the analysis and design activities simultaneously. The coding and testing were planned to be performed in parallel and we intended to begin testing the software five days after the programmers

[1] The term Version is commonly used for defining different versions of software rather than software parts. We will use the name Version throughout the thesis because it was the terminology that we used during the project.

begin to develop the code. We based our initial estimate on the desired functionality of each version. The dates displayed in the previous diagram were initial estimates, and they were modified as we received more precise definitions of the major software functions from the Analyst and Design Teams.

5.7 Resource Allocation

Resources are the personnel and equipment used in a project. Resource allocation is dynamic; it changes during the project's lifecycle. Initial personnel and material allocation is defined at the beginning of the project, based on our best estimates at that time. However, as the project unfolds, people leave the project, team size changes, new tasks are identified and allocation changes are needed.

The 34 students were divided into 10 teams: project managers(4), business managers(3), marketing managers(2), configuration managers(3), testers(3), knowledge managers(2), requirements analysts(6), quality assurance(4), designers(5) and programmers(2). All students, except three, had a second role as programmers. Of these three students, two had a second role as quality assurance members while the third had a secondary role as knowledge manager. To enhance the collaborative environment an effort was made to involve at least one member from each geographically distributed team in each of the above mentioned team roles. By pursuing this strategy, every distributed team had up-to-date knowledge in every area of the evolving project which led to constructive communication during team presentations. All members were made to feel that they could not accomplish the task without the help of their counterpart across the physical boundary. The list of resources used in the ieCollab project is presented in section 4.6.

5.8 Risk Identification

Risk can be defined as the likelihood of incurring problems or loss (Pressman, 1997). Risk involves uncertainty and a risk management plan is the tool used to quantify the uncertainty, to estimate the possible level of damage and to develop mitigation plans. The main objective of risk management is to develop a strategy with which to identify

potential risks and develop a contingency plan. In this section, we will describe how we identified the potential risks to the ieCollab project, how we estimated their probability of occurrence, how we ranked them by order of importance and how we mitigated them. We used the framework proposed by Pressman (Pressman, 1997) in order to identify the risks and estimate their impact on the project.

Risk Item Checklist	**Abbreviation**
Product Size Risks	PS
Business Impact Risks	BU
Customer Related Risks	CU
Process Risks	PR
Technology Risks	TC
Development Environment Risks	DE
Risk associated with Staff Size and Experience	SS
Performance Risks	PE
Support Risk	SU
Schedule Risk	SH

Table 5.4 Risk Itens

The following risk were identified in the ieCollab project:

Product Size Risks

There was little confidence in the software size estimate. However, the software to be developed was supposed to be small. We identified that there was a risk that if we began to create more functions than strictly necessary, delays in the project might occur.

Another aspect to cover here is related to the number of documents to be created. The number of technical documents and support documents that we planned to create was estimated to be large; we realized there was a risk that could create problems with storing and accessing documents in our Web-repository.

Business Impact Risks

The delivery deadline seemed to be reasonable in spite of the academic calendar and other project demands. Later in the project we realized the project scope was too large for the amount of development time that we had.

Customer Related Risks

As the ieCollab was an educational project, we did not have much customer-related risk. Our customer would be the faculty who was willing to participate in reviews and who did understood the software process. The only risk was the possibility that the software might not achieve the desired functionality. However, with increased customer interaction, the risk of not meeting the desired functionality seemed to us small.

Process Risks

In term of process risks, we took several measures at project preparation. For example, all team members took a one-term course on software engineering; the software standards were provided to all teams and everybody had access to documents from last year's Cairo project. We also scheduled formal technical reviews and walkthroughs and had a defined process of putting comments and reviews on the Website. The Quality Assurance team assured that the work done conformed to the IEEE Standards. In spite all these measures, we were able to identify the following risks:

- Having last year documents available to all was not a warranty that all were going to read and use them efficiently.
- The project management team was making use of software tools (Primavera/MS Project/ Vensim) to support the planning and tracking of activities. However, the team at that point was not experienced in using these tools, which created delays in creating schedules.

- There were no tools to support the testing process. We had to create our own testing tools as the project unfolded.
- We did not have well-defined quality or productivity metrics at the time of the elaboration of the project plan.
- We were going to work with different software languages and systems (Java, Oracle, Corba, other Websites, etc.). It was clear to us that it would difficult to ensure consistency among the different systems.

Development Environment Risks

The training and skills of team members were diverse. Some team members were more familiar with project management tools, while others are more accustomed to using code development tools. We tried to match the individual skills with the software development process as best as we could. The faculty asked the team members to choose roles in which they would feel more comfortable. It was very helpful, to have someone on team who had mastered the necessary skills and could help the others with deficiencies. Many development tools were available. For example, the project management software (Primavera) was available in its demo version.

Analysis and design tools such as Rational Rose were also available. The code generator (Java 2.0) was also available to all. However, at the beginning of the project we had the following deficiencies that would later cause development risks:

- There were no configuration manager tools available.
- The software tools are not integrated with one another. We had many tools, but we did not have a consistent development environment.

Technology Risk

The technology that we were going to use was not new. However, many team members had no previous experience in this kind of software development. The software package was going to integrate the existing Cairo system that was not fully tested at that time. Our software was supposed to interact with other commercial software (i.e. a calendar service), but at the planning phase we did not have a clear idea how to achieve this.

Risks Associated with staff size and experience

The majority of the team members had knowledge of Information Technology and Computer Science. However, we also knew that all team members were involved with other course projects and thesis work, which would reduce the amount of time available for the ieCollab project. At the planning phase we thought that the probability of people leaving the project was small. This assumption proved to be totally wrong; at the beginning of the second half of the project, a large number of students from CICESE left the project. We will discuss the causes and consequences of this in Chapter 9.

Other risks

- Performance Risk: ieCollab was being developed around the Cairo system. The Cairo project had not fulfilled its requirements. Hence, it could be expected ieCollab software would face the same difficulties as well.
- Schedule risk: The project would be finished at the end of the Spring term. There would be no schedule slippage; hence the quality of product would suffer if obstacles arose.

Risk Table

Following Pressman's framework for risk management (Pressman, 1997), after identifying all risks, we ranked them in terms of degree of impact (table 5.5) and probability of occurrence.

Degree of impact	Impact Values
Catastrophic	1
Critical	2
Marginal	3
Negligible	4

Table 5.5 Values for Risk Ranking

As we did not have sufficient data from the previous year, we used our best guess in defining probability of occurrence (table 5.6).

Risks	ID	Category	Probability	Impact
Confidence in software size estimate	1	PS	High	3
Small amount software reuse	2	PS	Medium	2
Large amount of documentation	3	BU	High	2
Large number of systems with which product should be interoperable	4	BU	Medium	1
Ineffective use of last year's documents	5	PR	Low	2
Inconsistency among different systems	6	PR	High	1
Lacking of training on tools	7	PR	Medium	2
Lack of software tools to support testing process	8	PR	High	3
Lack of quality or productivity metrics	9	PR	High	2

No configuration management tools available		DE	High	3
Software tools are not integrated	11	DE	High	4
No previous experience on software development	12	TC	Low	2
Use of untested Cairo routines	13	TC	High	1
Interface problems with another software	14	TC	Medium	2
Team members involved with other course projects	15	SS	High	2
People leaving the project	16	SS	Medium	2
Project doesn't meet all requirements	17	PE	Medium	1
Problems with Cairo features	18	SU	Medium	1
Quality effected by schedule slippage	19	SH	High	1

Table 5.6 Project Risks

Based on table 5.6, we grouped the risks that had the same probability of occurrence and the same impact (Table 5.7).

Risk ID	Probability	Impact
6,13,19	High	Catastrophic
3,9,15	High	Critical
4,17,18	Medium	Catastrophic
2,7,14	Medium	Critical
1,8,10	High	Marginal
5,12,16	Low	Critical
11	High	Negligible

Table 5.7 Ranking of Risks

Risks with high probability of ocurrence and with catastrophic or critical impact were our first priority. We began our mitigation plan by considering these risks.

Risk Mitigation

For each category defined in table 5.7, we developed mitigation strategies. The following section discusses how we planned to mitigate the anticipated risks. In chapter 8, we will discuss which plans were successful.

Strategy for mitigation of High Probability and Catastrophic Impact Risk

Risk 6- In order to assure consistency among different systems, we took the following measures:

- A Committee Board was formed from the leaders of each team. The plan was that committee would meet every week or whenever necessary to manage the changes that could affect other parts of the project
- If one team meeting dealt with information that was also pertinent to another team, one team member from the outside team would attend that meeting. This would help to keep consistency and coherency among the documents and plans.

- The Analyst and Design teams would agree on the common tools to be used at the beginning of the development. The suggestion was to use UML methodology.

Risk 13- Using untested Cairo routines

- We decided to assign one or two students to perform tests on Cairo programs. We decided to test all Cairo code, regardless of whether or not it had been tested the previous year. (However, this measure was not implemented; the Cairo code was so unstable that after some trials we decided to avoid using its routines at all).

Risk 19- Quality affected by schedule slippage

- We decided to do a critical analysis on the feasibility of the requirements. We wanted the requirements for Version 1 and 2 to be realistic, taking into account the short project duration and various student commitments outside of the project. We felt a small, reliable software package would be better than a large, unreliable one.
- If a schedule slip must be made we would reduce the requirements; we did not want to sacrifice quality.

Strategy for mitigation of High Probability and Critical Impact Risk

Risk 3- Overly large amount of documentation

- The Knowledge Management Team was asked to become active from the beginning of the development. There was a natural tendency for all teams to put every file on the web repository. However, we knew that putting so many documents in a disorganized way would increase the Quality Assurance team's workload substantially. It would be more efficient to put the final version of a given document On the Website, and store the draft versions somewhere else.

Risk 9- Lack of quality or productivity metrics

- The project manager team together with the quality assurance team decided to implement the PDCA quality assurance model and several metrics. We will discuss this in detail in the next chapter.

Risk 15-Team members distracted by work in other course projects

- The team leaders would give each team member individual responsibilities. The project management plan contained the general responsibilities of each team. The leaders of each team were responsible for monitoring the efforts of their team member in order to meet the team's responsibilities.
- Weekly meeting led by the project management team to discuss problems team members were facing with their academic schedule and other subjects. By so doing, the project manager team would be able to re-allocate resources to help teams complete tasks on schedule.

Strategy for mitigation of Medium Probability and Catastrophic Impact Risk

Risk 4- Product would not be interoperable with large number of systems

- As the programming activities would begin only after the analysis and the design were finished, the Programming team was given the responsibility of exploring the other systems that ieCollab would interact with (such as a database systems, for example).

Risk 17- Project fails to meet all requirements

- Review of Version 1 and 2 specifications. An overoptimistic approach could create unrealistic goals.
- Identify which features are indispensable and which are superfluous.

Risk 18- Problems with incorporating the Cairo System routines

The Cairo routines were not incorporated at all, as we discussed previously.

Strategy for mitigation of Medium Probability and Critical Impact Risk

Risk 2- Limited amount of software reuse

- The project manager team asked programmers to reuse code as much as possible in order to speed up the development process. There were several reliable development Websites with free code available.

Risk 7-Lack of training for tools

- Team members were asked to inform the project management team when they faced insufficient knowledge of the tool they were trying to use. The project management team contacted the faculty and scheduled a short course on the specific tool when it was needed.

Risk-14 Problems with the interface of another software package

- This risk could be mitigated by the programmer team's leader if he placed a member of his team in charge of figuring out how to interface with the outside software that ieCollab would interact with

Strategy for mitigation of Low Probability and Critical Impact Risk

Risk 5 – Not using last year documents effectively (Cairo project)

- The use of documents from last year's Cairo project would speed up the creation of manuals and certain documents. The Knowledge Management group was given the responsibility of making last year's documents available to all ieCollab members.

Risk 12-No previous experience on software development

- Placing an experienced team member in each team mitigated this risk. Most of the students had academic background in Information Technology and/or Computer Science programs, hence this risk was perceived to be low.

Risk –16 People leaving the project

- We tried to avoid this risk by splitting the leadership of the project between MIT and CICESE. We thought that keeping leadership distributed it would create motivation and morale would be increased. It did not work the way we wanted, however. We will discuss this issue in Chapter 8.

Strategy for mitigation of High Probability and Negligible Impact Risk

Risk 11 - No configuration management tools available

- The Configuration Manager Team was asked to develop a Web-repository with links to CVS software

5.9 Effort estimation

We used the Lorenz approach (Pressman, 1997) to estimate the number of key classes (basic unit in object-oriented programming) to be developed. Based on the amount of functions that each version would perform and the requirement analysis data at that point of the project, our initial estimate was that at least we would have to create 30 classes per version.

Estimated number of classes (IeCollab Version 1.0)	**Estimated number of classes (IeCollab Version 2.0)**	**Total**
30	30	60

Table 5.8 Number of Classes

In order to figure out the number of support classes, we multiplied the estimated number of classes by the empiric multiplier factor proposed by Pressman (Pressman, 1997).

Interface Type	Multiplier
No GUI	2.0
Text-Based user interface	2.25
GUI	2.5
Complex GUI	3.0

Table 5.9 Class Multiplier

Estimation for Version 1: Number of support classes =2.5*30 = 75

Multiplying the sum of key classes plus supporting classes by the average number of work-units per class (08 person days) to get the amount of person-days required (30+75)*8= 840 person-days. However, at that time we expected to re-use at least 60% of Cairo code, hence the estimation could be 336 person days. The programming of version 1 would involve 12 students for 31 days, or 372 person-days. For Version 2 we estimated 42 days. In order to improve our estimates, we planned to make use of the system dynamic rework model (Lynes, 1999). The system dynamics based estimates would be used only for the programming phase (Figure 5.3).

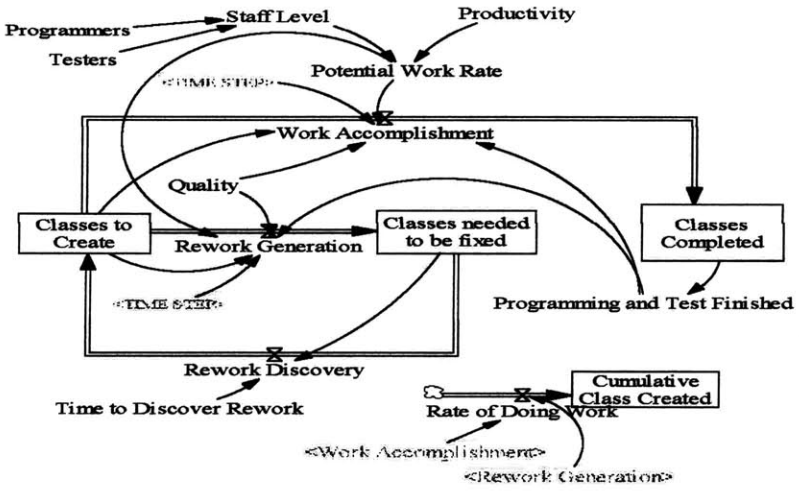

Figure 5.3 The Rework Model (Lynes, 1999)

5.10 Creating the WBS and the Project Schedule

The WBS defines the work to be done. Figure 5.4 shows the WBS that we used, the teams members responsible for performing each task and the due dates.

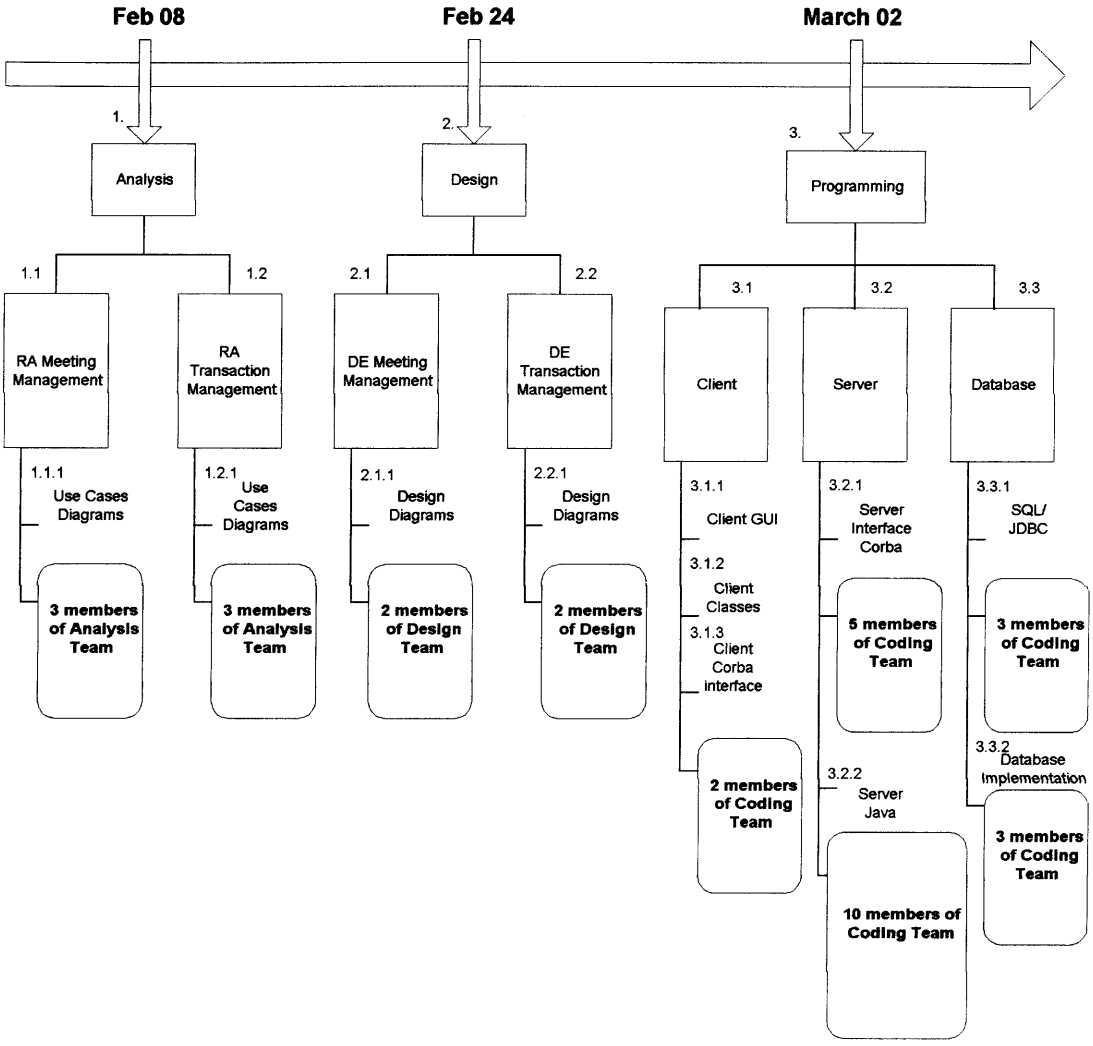

Figure 5.4 Work Breakdown Structure

Once we had defined all activities, estimated the duration of each activity and allocated resources, we were ready to create the project schedule. The project schedule

shows the milestones and checkpoints, the deliverables at the milestones and the team members involved. The project schedule suffered several modifications and adjustments while the project unfolded; we will discuss this in the project adaptation chapter. Table 5.10 shows the initial schedule for our project.

Initial Project Manager Plan			
Date:	Event:	Documents Involved:	Team Involved:
11-Nov-99	**Project Start**		All teams
18-Nov-99	Walkthrough of PM and BM Plan	Project Manager's Plan v1.0. Business Manager Plan v1.0	All teams
19-Nov-99	Definition of General Project Goals/Functions	Project Manager's Plan v1.0.	Team leaders
7-Dec-99	Walkthrough of Requirement Analysis v1.0	Requirement Analysis Documents	All teams
18-Jan-00	Feedback on initial activities	Comments on all documents	All teams
18-Jan-00	Analysis and Design Version 2 Begins Programming Version 1 Begins		Analyst, Designers, Programmers
23-Jan-00	Test Version 1 Begins		Tester
01-Feb-00	Project Management team Presentation	Project Manager Documents	All teams
08-Feb-00	Requirement Analysis team Presentation	Requirement Analysis Documents	All teams
15-Feb-00	Design team Presentation	Designers documents	All teams
18-Feb-00	Programming Version 2 Begins		
22-Feb-00	**Delivery Version 1.0** Programming **Version 2.0 begins**		
23-Feb-00	Test Version 2 Begins		
29-Feb-00	Programmers team presentation	Programmers document and code	All teams
07-Mar-00	Quality Assurance team presentation	QA Documents	All teams
14-Mar-00	Configuration Manager team presentation	CM Documents	All teams
28-Mar-00	Knowledge Manager team presentation	KM Documents	All teams
04-Apr-00	**Mock-up of final presentation**		All teams
11-Apr-00	Final Presentation **Delivery Version 2**		All teams

Table 5.10 Initial Project Schedule

5.11 Project Measurables

In order to determine whether or not the project was following the schedule, it was important to define a way to measure the project's progress. In our initial plan we defined 10 checkpoints (Table 5.11).

Checkpoint	**Date**	**Week**	**Project Time spent (%)**
1 -Elaboration and approval of the project management plan.	18/Jan	05	29%
2 -Version 1 and 2 specifications completed and approved.	03/Feb	07	41%
3- Analysis and Design of Version 1 completed and approved.	17/Feb	09	53%
4 -Approval of Version 1 Code and Test Cases. Delivery of Version 1.0	22/Feb	10	58%
5- Analysis and Design of Version 2 corrected and approved.	02/Mar	11	64%
6- Configuration Manager Plan approval.	16/Mar	13	76%
7- Mock-up of final presentation and DEMO	04/Apr	16	94%
8- Version 2 code completed and approved. Test cases approved.	06/Apr	16	94%
9- Checkpoint 9: Final Presentation Delivery Version 2	11/Apr	17	100%

Table 5.11 Checkpoint Plan

At this point we did not have a clear idea of the mechanism that we would apply to measure the progress. We only knew what should be accomplished at the checkpoints

in order to complete the project by deadline. We will explain the structure that we used to measure the project's progress at the checkpoints in next chapter.

5.12 Defining the reward structure

The main driver of work effort is the team's motivation. To have a good plan and a good control mechanism is not sufficient to ensure that the project will be developed the way that was planned. If the people involved in the project do not have the necessary motivation, all project activities can be affected. It is necessary to keep the project members motivated and committed to achieve the project goals. If the people involved do not receive constant incentives, morale will decrease. Many companies nowadays use stock options as incentives. However, the incentives need not necessarily be financial; recognition by peers and superiors is also an incentive. The incentive can be also in terms of professional growth opportunities. In the ieCollab project, keeping the teams motivated was a constant challenge. The project was complex, the schedule was tight and the coordination with virtual teams was difficult. To make things worse, at middle of the project many students from CICESE left, which increased the workload for those who remained. Furthermore, many students were taking other courses simultaneously and/or working on their theses. This increased the workload and decreased motivation. We did not really have a reward system; the only thing close to an incentive was the final grade. However, how the grade would be given was not defined; it was not clear if the grade would be based on individual effort or on team effort.

5.13 Summary

In this chapter we covered the most important aspects of project planning, describing the main issues of project planning by means of a example of a real project. Special emphasis was given to the risk assessment and mitigation. In the next chapter we will discuss the main issues involved in project execution.

Chapter 6

Project Execution

Project execution is composed of project monitoring, control, quality assurance and creation of a log of lessons learned (Figure 6.1). In this chapter we will discuss each of this components in detail we will describe how we dealt with the main issues in the ieCollab project.

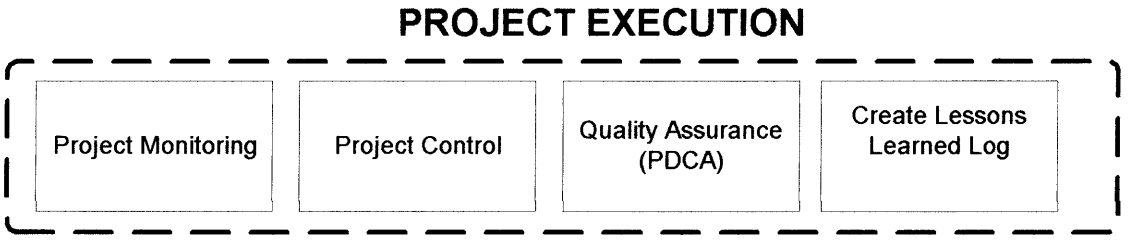

Figure 6.1 Project Execution Activities

6.1 Project Monitoring and Control

Project monitoring can be understood as the set of actions taken in order to determine whether or not the project development is following the plan. Project control is the measures taken to correct the deviations between the actual project and the planned one. Project control is directly related to the project monitoring once the information obtained from the monitoring process is used to control. There are two sets of measures

currently used in project monitoring/control: cost-related measures and engineering work-related measures. Both approaches are complementary. The cost approach involves the Earned Value Analysis, which is outside of the scope of this thesis. There are several books that cover Earned Value Analysis. Among them we could suggest Shtub's book on project management (Shtub et. al., 1994). In this chapter we will focus on controlling and measuring engineering work. We will discuss the control measures taken in the ieCollab project as example.

Figure 6.2 shows a simple block diagram representing the relationship between project monitoring and control. The desired project level comes from the project management plan. The project monitoring process compiles many different sources of project reports into a single document that gives an idea of the actual project status. The project manager can control the project by adding more resources, realigning resources, changing schedules and changing scope. The results of these actions on the project are also monitored and this process continues throughout the project's duration.

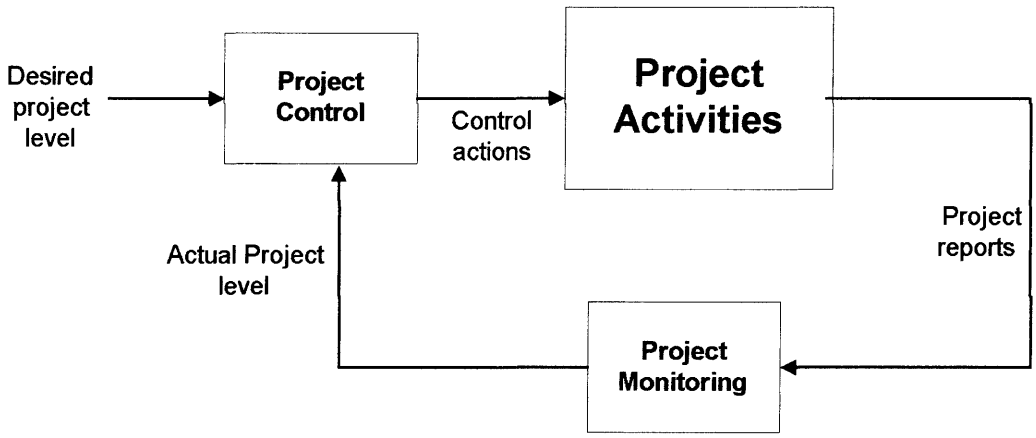

Figure 6.2 Relationship between Project Monitoring and Project Control

The project plan makes clear project's main objectives, tasks and responsibilities. There are many objectives to watch for; consequently, it is necessary to determine which are key and to develop mechanisms to keep track of whether they are being met. The objectives should be realistic and quantifiable, and the people involved in the project

should be committed to achieving them. In order to do so, checkpoints are created in order to help verify if the project is developing as planned. Monitoring is directly related to the measure of progress at these checkpoints. If you cannot measure the progress, you cannot control the project (Thamhain, 1992). There are several issues related to the control and measure of engineering work. In the rest of this chapter we will focus on the most important ones: project visibility, project complexity and work effort. We will also discuss the role of quality in project management activities and finally we will discuss the monitoring and control activities of the ieCollab project.

Ideally, the project manager would have total visibility of the project. However, this is not possible to achieve. In fact, what the project manager sees is only snapshots of the project at certain points in time (McConnell, 1996). The top plot of figure 6.3 shows the ideal project visibility. Ideally the project manager should know the current status of the project at any point in time. However, it is not possible to achieve this in most of the cases. Usually the project manager defines ways of monitoring and quantifying the development of the project. This can be done by weekly reports, for example. The bottom plot of figure 6.3 gives an idea of the visibility achieved by using weekly reports. The project manager cannot know every detail of what happened during the week; this is represented by the covered parts of the plot. The project manager will have only a snapshot at the end of the week, and based on this, he can have an idea of project's progress.

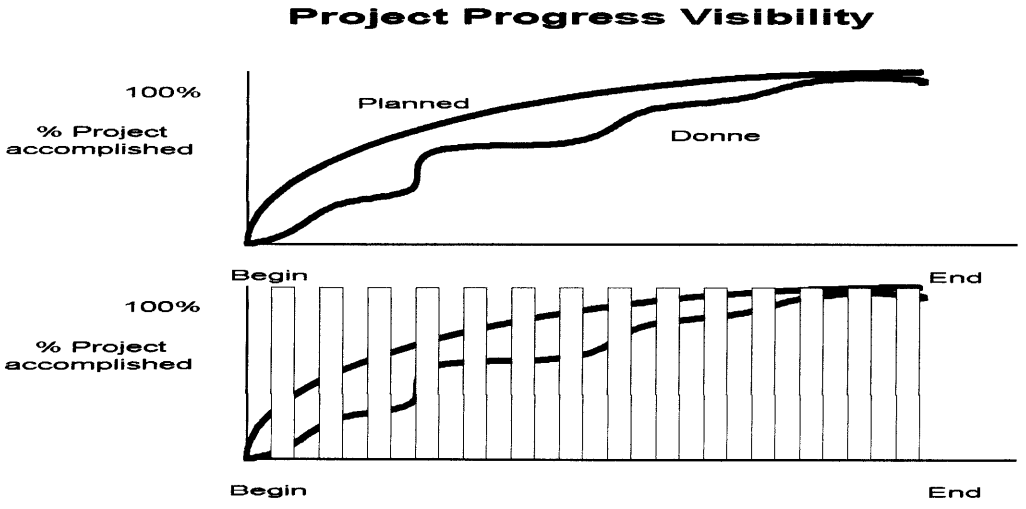

Figure 6.3 Project Visibility (McConnell, 1996).

Most projects are complex; it is impossible to plan every single activity involved in the project at the beginning. The project manager creates a plan based on what the team knows about the desired product at the beginning of the project. As the project unfolds, the teams can discover that new activities should be added to the project and give this feedback to the project manager. The technical difficulties become clearer and the project manager must take fast action in order to overcome them. These control actions can be training, acquisition of new equipment or software tools, reallocation of personnel, and so on.

The project manager has to be aware of the necessity of taking control actions rapidly. If the action is related to re-allocation of personnel, it is important to match personal skills with the task's difficulties. If the person assigned to do a task does not have the necessary skills, the project manager has to provide the necessary support (training, short-courses) in order to help him or her accomplish the tasks.

Taking the ieCollab project as an example, at the beginning of the project, the project manager team knew the basic requirements given by the business manager and marketing manager teams. We knew the basic features that our final product would have, but we did not have a clear idea of many technical aspects were involved. We knew what to do, but we did not have a clear idea of how to do it. For example, we knew that the software product would have three main components-- the Client component, the Server component and the Database component-- but we were not sure about the tools that we are going to use to build each part. As the project evolved, the Requirement Analysis team and the Design team described the system in detail and recommended the interfaces to create and the tools to use. The Client component was divided in three parts: Client Graphical User Interface (GUI), Client Classes Java and Client CORBA Interface. The Server component was divided in two parts: Server CORBA Interface and Server Classes Java. Finally, the Database component was divided into Interface JDBC/SQL and Database Oracle(Figure 6.4).

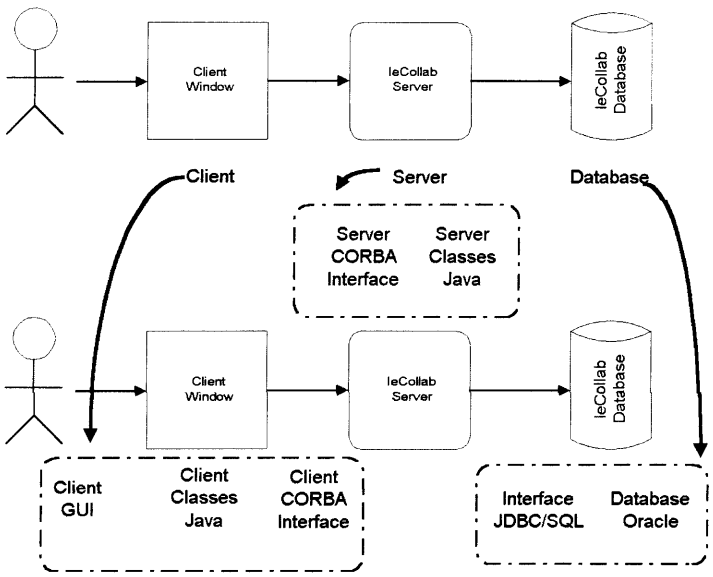

Figure 6.4 ieCollab Architecture

Dividing the programming into parts made it easier to understand and simpler to manage. Once the project was divided in parts, we divided the programmers in five teams. We tried to make the division in order to match the individual skills of every team member with the appropriate task.

The figure 6.5 shows the final ieCollab programming structure. Dividing the programmer team into small groups helped us to have a better control of the group's activities. The team leaders of each group could assign individual programming tasks to each team member. The top part of Figure 6.5 shows the amount of functions developed by each programming team and the number of students on each team.

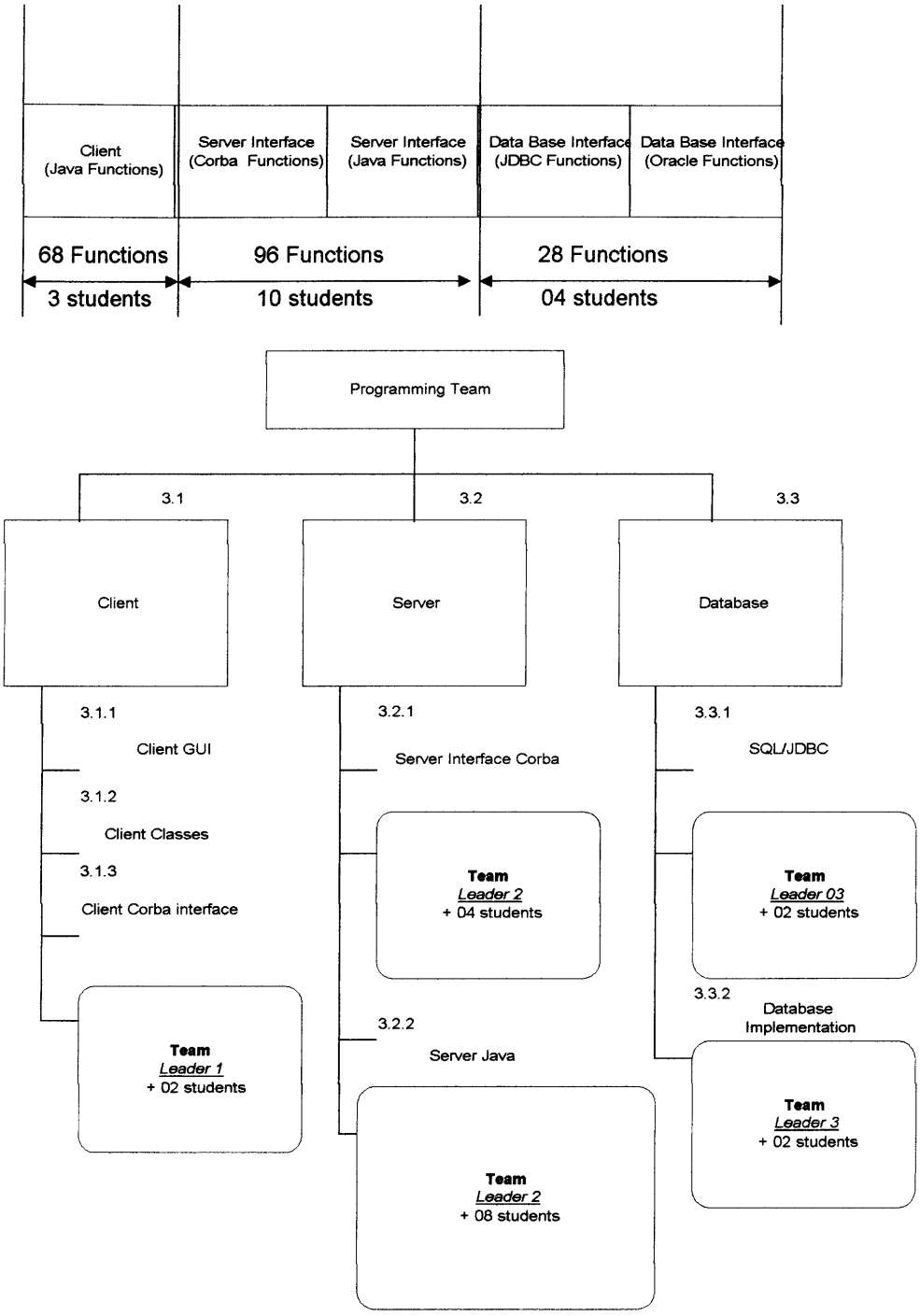

Figure 6.5 ieCollab Programming Structure

6.2 Quality Assurance

In order to reduce the amount of reworking, the quality of the project must be high. The most used method to assure the quality of the product development process is the PDCA (Plan, Do, Check, Act) model.

We will describe the basic ideas of PDCA (Clausing, 1994); (Shiba et. al.1993), and we will give an example of how it was used in the ieCollab project. PDCA is an on-line quality control model composed of four sequences of actions defined by the verbs Plan, Do, Check and Act.

The "Plan" part of PDCA is the definition of goals and targets. In the ieCollab project, the project management team created the overall plan. The "Do" part is related to the work developed. In the ieCollab project, all teams were working at all times. The "Check" part is the project monitoring activities and the "Act" part is the project control activities. In the ieCollab project the check activity was done together by the project manager team and quality assurance team. The team's weekly report was the tool used to check if the project was following the plan. Based on these reports, the project management team took two different actions (the "Act" part of PDCA) every week: feedback actions and forward actions (Figure 6.6).

The feedback action was related to the improvement of the development process and to the improvement of the plan for the next product. Based on what the team leaders highlighted as their problems, the project management team could, for example, allocate more personnel to help the teams or schedule lectures on specific topics. The weekly report also brought in suggestions from all teams. These suggestions were also recorded in order to improve the next year's project plan. We will see how this was useful in Chapter 8.

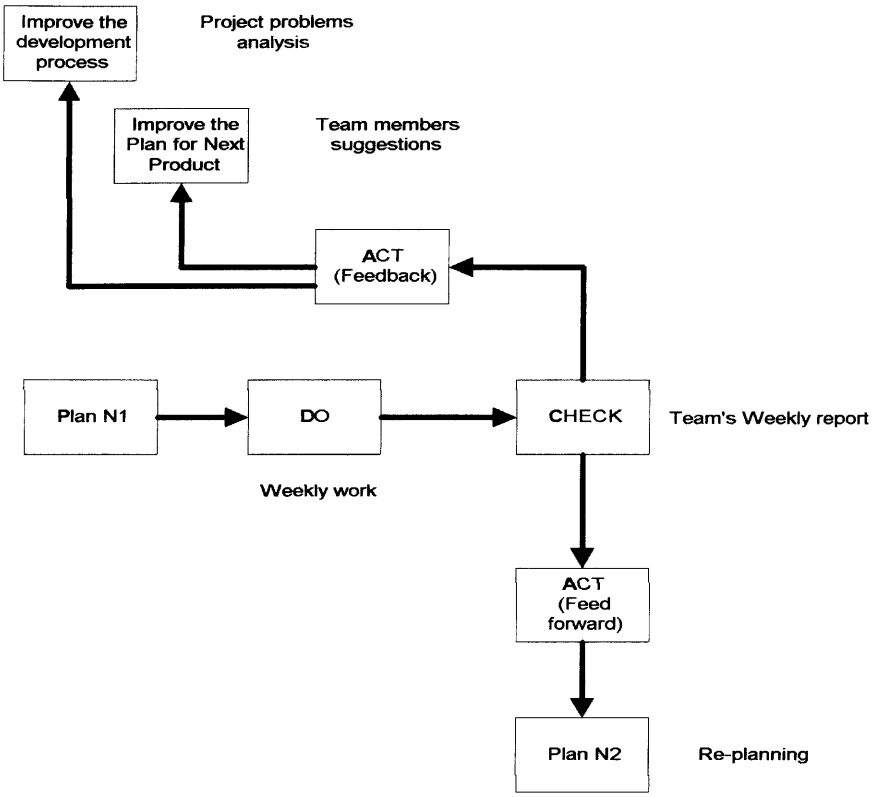

Figure 6.6 PDCA Application (Clausing, 1994)

The forward action was related to modifying the plan. During the lifecycle of the ieCollab project several modifications were made on the original plan.

6.3 The ieCollab Monitoring/Control Process

In order to monitor the project, the project manager team asked each team leader to submit a weekly report on the activities of the team. In these reports they stated the tasks they had to do, the percentage of task accomplished, the problems they faced; they also made suggestions. Based on this information, the project manager developed a weekly project control report and presented it to all teams. The idea of this report was to

show to all team members the project's status, the problems that each team was facing and the measures taken by project managers to reduce these problems. It was also an opportunity to collect more information and reach consensus on controversial aspects of the project.

The project manager's weekly report was basically done in form of a presentation to the team (MS PowerPoint presentation). The presentation was planned to take no longer than 15 minutes and to cover the topics shown on table 6.1 (Thamhain, 1992).The presentation covered not only the project control and monitoring, but also project adaptation. In the rest of this chapter we will focus on the first row of table 6.1. The aspects covered in the rest of the table will be discussed in the next chapter.

All presentations were planned to have some slides done and presented by the Quality Assurance team. The idea was that the project had to evolve following the plan but with quality. The project management team and quality assurance worked together in order to make this presentation useful to all teams.

Project Manager Weekly Report Topics	**Tools used**
Work status tracking and analysis	▪ Activities Status ▪ Project Status ▪ Team accomplishments
Changes introduced	▪ Network and Gantt charts
Problems and their impact	▪ Cause effect diagrams ▪ Solution matrix
Progress made	▪ Summary of team's activities
Open items	▪ List of items to discuss

Table 6.1 Topics and Tools for Weekly Report

Based on the team leaders reports the project manager developed three plots to show the activities status, project status and the teams' accomplishments. The first (Figure 6.7), the Activities Status plot, compares the percentage of accomplishment of each task on the critical path, the planned with the actual. This plot shows only the activities on the critical path; the project support activities are not shown.[1]

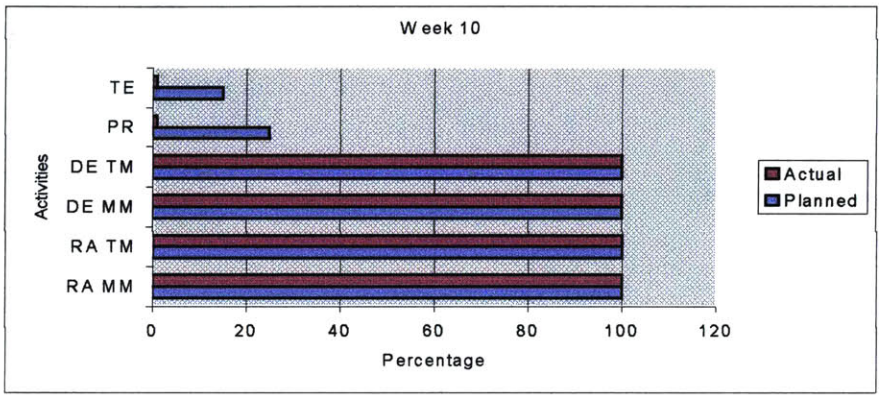

Figure 6.7 Activities Status

The second plot, Project Status (Figure 6.8), compares the percentages of the whole project. This plot gives a clear idea as to whether or not the project is developing according to the schedule. It also shows the checkpoints.

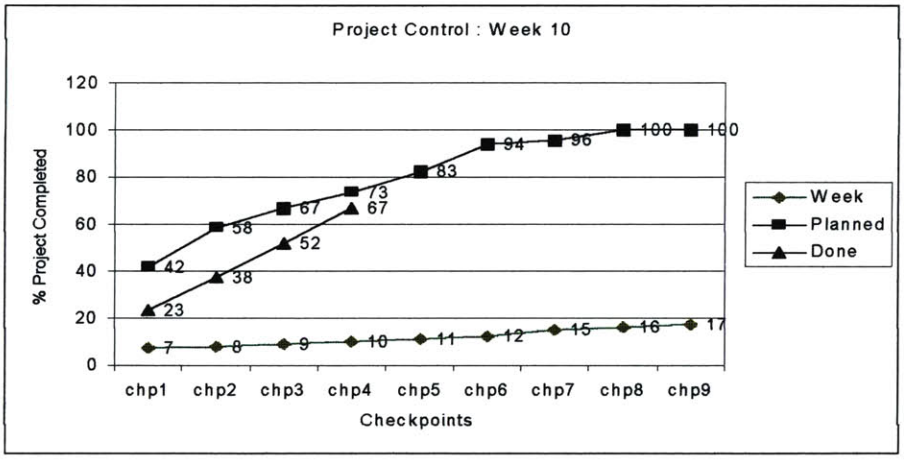

Figure 6.8 Project Status

[1] TE means tester group, PR programmers, DE TM designer transaction management, DE MM designer meeting management, RA TM requirements analyst transaction management and RA MM requirement analyst meeting management

The third plot, the Team's Accomplishments, was more subjective (Figure 6.9). It was based on the project management team assessments of what the groups did during the week compared with what they were expected to do.

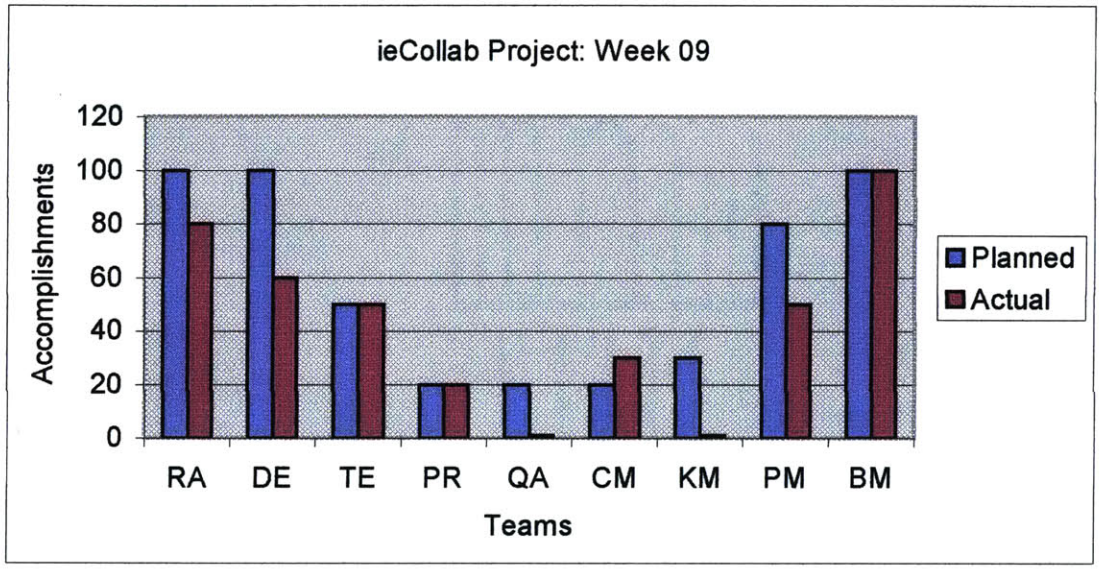

Figure 6.9 Teams' Accomplishments

6.4 Creating a Lessons Learned Log

The creation of a lessons learned document was a natural sub-product of the project execution. We will discuss the lessons learned of ieCollab project in the chapter 8.

6.5 Summary

In this chapter we discussed the monitoring and control processes and the importance of the quality assurance process. We showed how we applied the Thamhain (Thamhain, 1992) framework in the ieCollab project. We also used the PDCA model to improve the quality of both the final product and the development process.

In the next chapter we will cover project adaptation issues. We will continue the explanation of the Thamhain framework, this time focusing on the changes introduced, emerging problems and their impacts, and the progress made (see Table 6.1).

Chapter 7

Project Adaptation

In this chapter we will discuss the main issues of project adaptation, following the model that we described in chapter 3 (figure 7.1). Our approach will be by means of examples of how the adaptation process was applied to the ieCollab project.

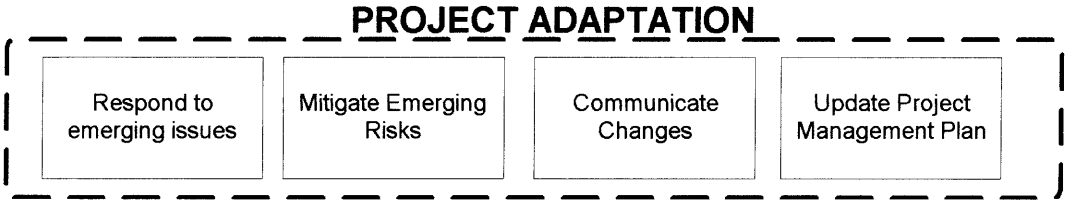

Figure 7.1 Project Adaptation Activities

7.1 Responding to emerging issues

As mentioned in the last chapter, at the beginning of the project the project manager and all the team members lacked a clear idea of all technical aspects of the project. As we stated in the previous chapter, in the ieCollab the project manager's report informed all the team members of the changes introduced (see table 6.1). The idea was to change in order to respond to emerging issues. Each time a substantial change occurred,

all team members were informed. Figure 7.2 shows an example of information addressed concerning the change in the project's network of activities.

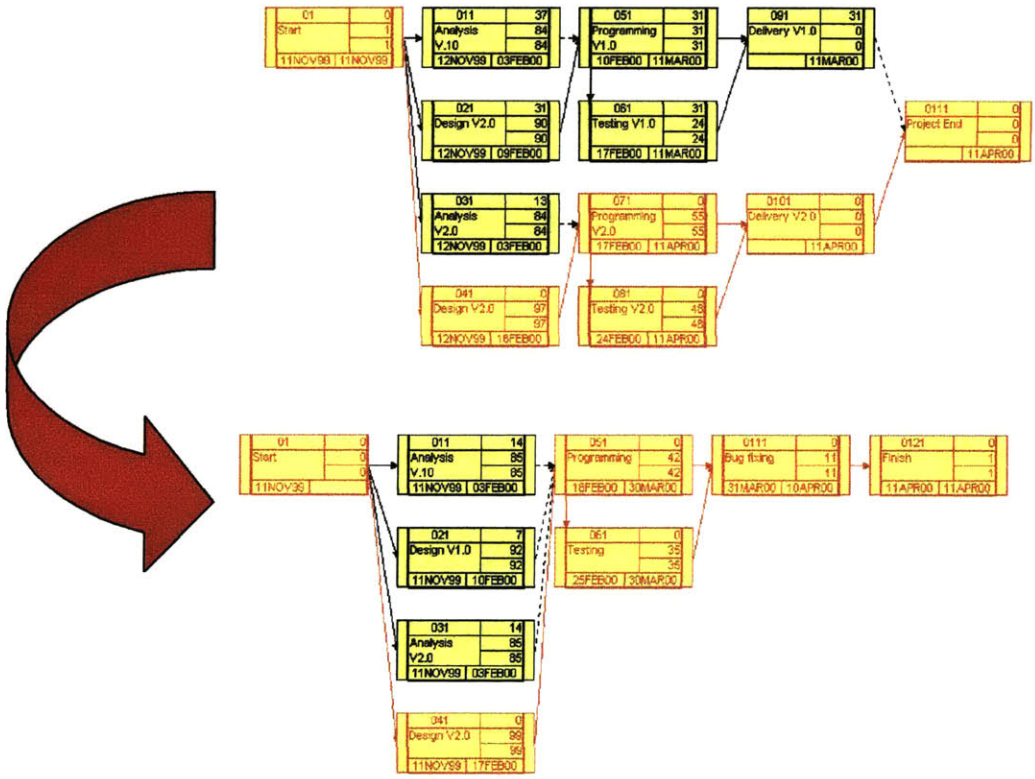

Figure 7.2 Project Change Example

This figure was used to show to all team members that the project management team, based on suggestions from designers and programmers, had decided to change the network of project activities. The designers and programmers had pointed out that it would be better to program the meeting management and the transaction management functions together because it would reduce the integration problems. This suggestion was accepted by the project management team and the modification was presented to all team members. Note that the suggestions from the designers and programmers was based on a better understanding of the technical issues; as the level of understanding increased, the level of planning became more accurate.

7.2 Mitigating Emerging Risks

Every week the team leaders sent the PM team a report on the amount of tasks their team had to do, what was possible to accomplish, problems and suggestions. Based on this information the project management team ranked the week's emerging risks and worked in order to find a solution to them. Table 7.1 shows one example of a week's problem and issues raised by the teams.

Week 09 Problems	**Issues raised**
(1) Poor project control	▪ Inadequate feedback from team leaders ▪ Ignorance about the team problems ▪ No coordination and control over CICESE team members ▪ Poor leadership and ability to manage conflicts ▪ Little involvement of teams leaders on planning/control ▪ Lack of interest of team members on the schedule ▪ Difficulties on mitigate risks, even those already identified
(2) Poor communication	▪ All sort of communication problems between remote teams and local team (MIT team x CICESE team), between teams (Analyst team x Designer team) and between team members
(3) Technical difficulties	▪ Difficulty to define the work in details ▪ Lack of knowledge in data base issues (Corba, JDBC, Client/Server) ▪ Many last minute changes in specifications
(4) Lack of motivation	▪ Work not perceived as important, feeling of no professional growth ▪ Many from CICESE/MIT students left the project ▪ No reward system
(5) Lack of commitment	▪ Indifference and apathy of many team members
(6) Conflict and confusion	▪ Small mutual trust ▪ Small cooperation among teams ▪ No control of team leader over the teams ▪ Inefficient sharing of knowledge

Table 7.1 Emerging Risks

Based on the information available, the Project Management team together with Quality Assurance team developed the cause-effect diagram in order to figure out the root causes of the problems (Figure 7.3). The cause-effect diagram was the quality assurance tool used to determine the main causes of the problems.

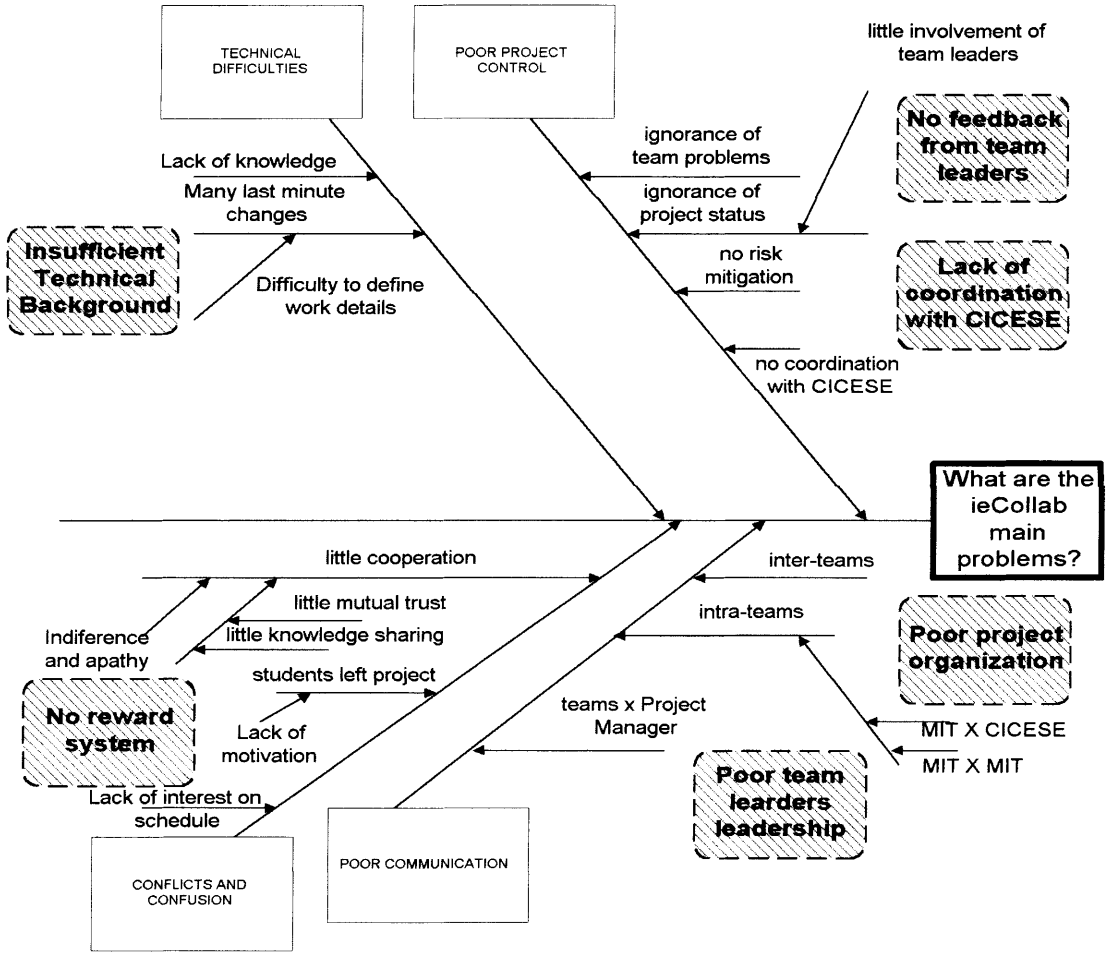

Figure 7.3 Cause-Effect Diagrams (Amaral et. al., 1999)

The final step taken by the project management team was to create a solution matrix (Table7.2). The table summarizes the measures taken by the PM team in order to help to address the emerging risks. This table shows the root problems, the team members who were chosen to address these problems and describes the way the problems will be solved and by what dates.

Root Problems	Who	What	When	How
Insufficient Technical Background	Student A and student B Faculty	▪ Lecture on CORBA/JDBC ▪ Include more technical lectures on Fall term and IAP	▪ Feb 24 ▪ Next year	▪ Lecture to all team members ▪ Include in next year course calendar
No feedback from team leaders	Project manager A	Enforce team leaders' sending the weekly reports	Every Monday by noon	By email
Lack of coordination with CICESE	Project manager B	Keep weekly contact with CICESE	Every week	By email or telephone
Poor team leaders leadership	Team leaders	Become more involved	Every day	Send email copy of all messages between teams to PM
Poor project organization	Project management team/ Faculty	Include the "interface" team members	Suggestion for next project	Include in the project organization

Table 7.2 Solution Matrix

Tables 7.1 and 7.2 give a snapshot of the problems that occurred in week 9 and the corrective actions taken to solve them. Showing these tables to the teams revealed to all that the project management team was aware of the problems and was taking measures to solve them. It was a clear message to the teams that their weekly reports had been read and that their recommendations had been considered in searching for solutions to the project's problems.

7.3 Communicating Changes

All changes introduced in the ieCollab project management plan were discussed with the teams during our weekly meetings. The remotely located teams participated in most of our meetings, sometimes using communication software and sometimes using the telephone. The solution matrix, the cause-effect diagram and the project schedule were posted on the wall of our meeting room. We also posted the changes introduced on our Web-repository, in the project management folder.

7.4 Updating the Project Management Plan

The use of the PDCA methodology that we followed helped us to update the project management plan when it was necessary. The schedule was the part of the plan that changed more than anything else. In order to keep every team member informed about the changes, every week the project management team sent a message to all team members, making clear the changes that were being introduced.

7.5 Summary

In this chapter we covered several aspects of project adaptation. We showed the importance of keeping all team members aware of project changes and the importance of mitigating emerging risks. We gave an example of how useful quality assurance tools were, specially, the cause and effect diagram and the solution matrix. In the next chapter we will discuss the lessons learned in the ieCollab project.

Chapter 8

Lessons learned from ieCollab Project

The post-project learning helps the project manager figure out the successes and the failures of the actual project. The lessons learned from a finished project will benefit not only ongoing projects but future ones (Meredith, 1989).

In this chapter, we will analyze the achievements and mistakes of the project management team during the life of the ieCollab project.

8.1 ieCollab Project Preparation Characteristics

The ieCollab project was an eight-month educational project. The initial preparation phase was aimed at developing students' skills on the software development process; the second phase was the project itself. The initial phase, project preparation, took the whole fall term of 1999. It was based on lectures on software engineering and the software development process. During this period, all students became familiar with the roles that they would take in the next term period (Spring 2000). Table 8.1 presents an overview of the actions taken in project preparation and the main problems encountered.

Actions taken	Problems
1. Lectures on software engineering and the software development process.	Technical programming issues were not covered. It was assumed that this information was provided in other courses.
2. Every week a different professor from different university presented the lecture	Lack of attention, interest and motivation when professors from other universities conducted the lecture.
3. The lectures were held simultaneously in MIT-USA/CICESE-Mexico and PUC-Chile using web resources and software communication tools	Many communication problems occurred. The software we used for communication was not reliable; many times we face audio and video problems. It also took a long time to establish connection with other universities.
4. Students work on team-based assignments in order to become familiar with all roles of the project	Insufficient time to cover all aspects of the software development process.
5. The faculty imposed the organization for the project. However, the students were free to choose among different roles.	The project organization structure was good; however, it can be improved.

Table 8.1- ieCollab Project Preparation Problems

During the fall term of 1999 the students became familiar with the basic principles of the software development process. Emphasis was given to the basic techniques of analysis and design. This was very helpful. However, the subsequent project suffered from a lack of students' expertise in programming issues (knowledge of programming languages as Corba, JDBC, for example) and databases principles. The basic assumption of the faculty was that these issues were being covered by other courses. However, this assumption was not correct; the other courses did not cover these

topics in the depth needed. This problem could have been attenuated if we had had lectures on these programming and database issues during the IAP period (winter term). The faculty also knew that some members of the project had knowledge on these topics and assumed they would disseminate this knowledge to other students. This assumption also proved to be overoptimistic; the students who had this knowledge were involved with many other courses and activities, and had trouble sparing the time to teach others.

Another interesting point to cover is the lack of effectiveness of lectures given by professors from remote universities. The use of Web communication for real-time lectures was not very effective. A professor could not see what all students were doing while he/she was presenting the lecture. Sometimes students were working on assignments for other courses, for example. Had we used videoconferencing, this problem could have been reduced. Furthermore, manner in which the lectures were conducted by remotely located professors, presenting many slides and asking for questions at the end of the presentation, was very inefficient. The students lost focus easily and at the end of presentation we were too tired and bored to ask questions.

The use of Web resources for these real-time lectures was also not efficient. It was usual to lose contact with Mexico and PUC; often we could not hear or see each other. This situation generated frustration and lack of interest. It would be better to have the lectures from remotely located professors recorded on tapes that students could watch as an assignment. In this way, we could use the real-time web lecture only to ask the professors questions; it would be much more helpful and interesting.

These lectures were, however, helpful in giving us an idea of the kind of the problem that we would face during the project. It was impossible to cover all aspects of the software development process.

Another issue to address is it that we did not spend much time reviewing the Cairo Project documents from the previous year. There were no formal discussions of these documents, although several groups consulted these documents on their own initiative. The project manager group read carefully the previous year's project management plan. This was very helpful; many lessons were learned. If all teams had read last year's plans and documents we would have avoided making the same mistakes.

We also did not define the project's scope or metrics with which to measure its success at the beginning of the ieCollab project. The scope definition was somehow established in the Business Manager/Marketing manager plan. However, the project's scope changed a lot during the project's lifetime. The Business Manager/Marketing Manager plan defined the desired features of the software in a very ambitious way. By the middle of the project it became clear that it would be impossible to develop all desired functions in the time that we had. Taking into account that the ieCollab is a continuous educational project and every year teams spend a whole term (fall term) learning the basics of project development, it would be useful to have a general framework (like the one presented in figure 8.1) that defines the scope of each ieCollab Project Version.

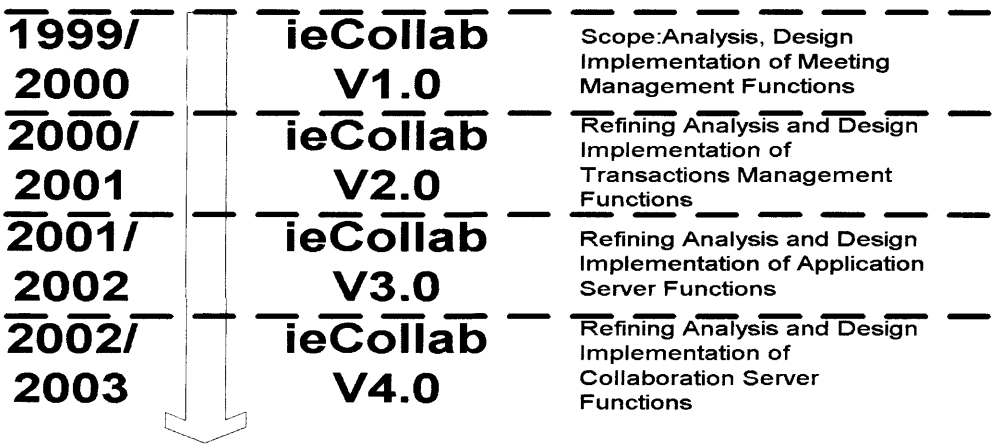

Figure 8.1 Proposed Versioning Framework

8.2 ieCollab Project Organization

The 34 students were divided in 10 teams: project managers (4), business managers (3), marketing managers (2), configuration managers (3), testers (3), knowledge managers (2), requirements analysts (6), quality assurance (4), designers (5) and programmers (2). All students except three, had a second role as programmers. Of these three, two had a second role as quality assurance members; the third one had a secondary role as knowledge manager. This organization proved to be not as efficient as planned. The ieCollab project faced serious communications problems among the teams. If in the beginning of the project we had had more students who had second roles different than programmer the communication between teams could have been much better. Students with dual roles could act as "link elements" (figure 8.2). The main role of these students would be to participate in meetings of both teams and make sure to inform one team of what the other team was doing.

For example, if we had on the Requirement Analysis team one student who had a second role as designer, the communication problem between these two teams would be eliminated and the analysis would be much more consistent and coherent with the design. The same thing applies to programmers and designers and between the Knowledge Management and Configuration Management teams. The communication was good between the Project Management team and the Quality Assurance team and between the Design team and the Programming team, because on the Project Management team we had one student whose second role was quality assurance. Furthermore, all designers had second roles as programmers. Another important aspect to define in the beginning of the project is who the team leaders will be and who will take the team leader place in his/her absence. However, merely defining who the team leader will be is not sufficient; it is also necessary to train these individuals in notions of leadership and clarify their responsibilities as team leaders.

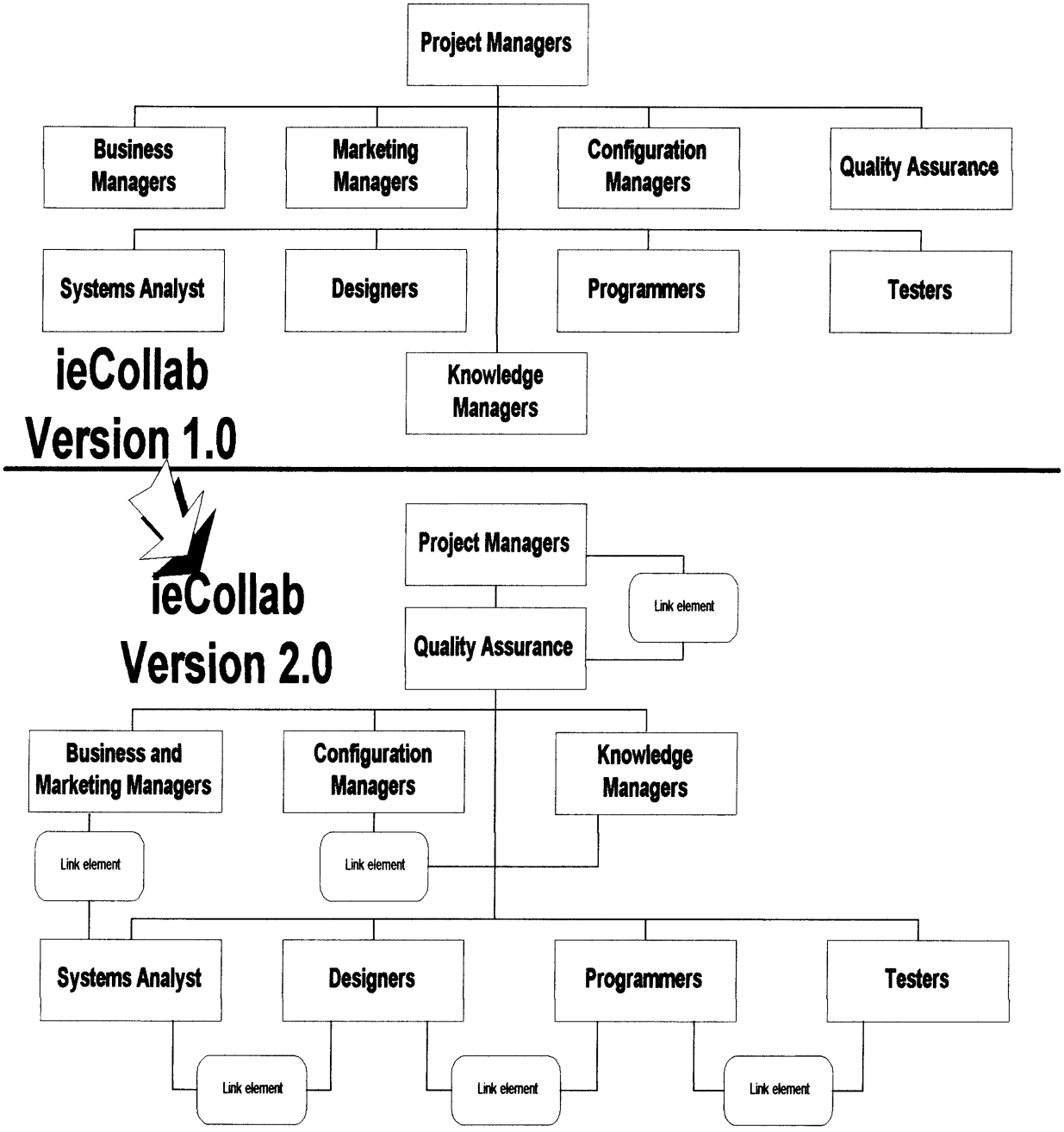

Figure 8.2 Proposed Project Organization

8.3 ieCollab Project Planning

The ieCollab project management's plan began at the end of the fall term of 1999 and was fully developed during the IAP period (winter term, one month in length). At the beginning of the Spring term the plan was presented to all team members.

There were some mistakes during the planning process:

- The project manager team did not enlist the participation of other team leaders during the elaboration of the plan. The initial plan was practically imposed. However, one week after the plan was released we had some feedback from team members and we introduced modifications based on this feedback, mainly in the schedule. If team members had participated more in the elaboration of the plan, many problems that we faced during the development of the project would have been reduced. For example, sometimes one or another team member said that he/she was not sure about his/hers responsibilities. If the team members had participated in the elaboration of the plan, they would not have had questions about their roles.
- We rushed directly into the project, without spending time discussing the project management plan. The plan was presented, we had not feedback from teams, and probably most of team members had not even read the plan. However, the project manager team at least made the schedule visible to all. The project manager team used a large calendar fixed to the wall of our project meeting room. All activities and milestones were written on this calendar. We also sent a message every week to all team members reminding them about the next week's activities. This helped to keep everyone updated on the project manager plan.
- The allocation of resources was based more on our best guess than on accurate estimation of effort for activity. At the beginning of the project we did not have a clear picture of the complexity of the product; as the project developed, the estimation became more accurate and we began to use system dynamic model to have better estimates. For example, at week 13 of the project, the programmer's team leader divided the programming team into three sub-teams: Client Team, Server Team and Database team (Figure 8.3).

The server team identified 96 functions to create and proposed delivering it in 20 days. This was based on his gut feelings, not on any formal estimate.

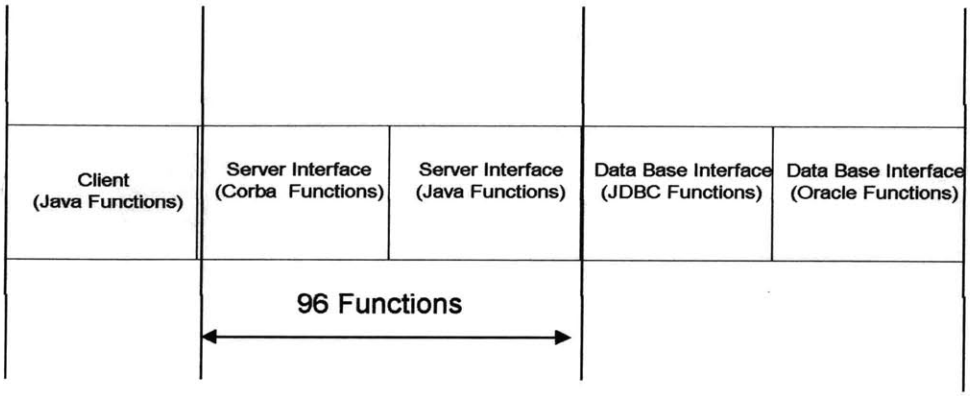

Figure 8.3 Programming Functions

Based on the number of functions to implement and the teams' size, the project management team used the system dynamic re-work model (Lyneis, 1999);(Hines,1999) to determine if their estimate was accurate (Figure 8.4).

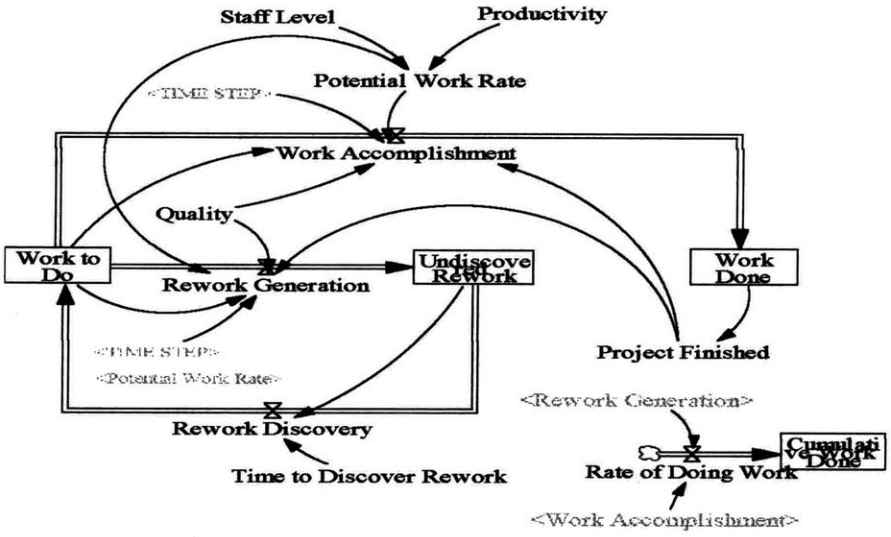

Figure 8.4 Rework Model (Lyneis, 1999); (Hines, 1999)

The project manager team analyzed three different scenarios: optimistic, more likely and pessimistic. In the optimistic scenario, we estimated that each team member would implement at least one function/day. In the more likely scenario we estimated that each programmer would implement one function each couple of days and in the more

pessimistic scenario we estimated that each programmers would spend 4 days implementing each function (Table 8.2).

Scenarios	Productivity	System Dynamic Estimate
Optimistic	1 Function/day	11 days
More likely	0.5 Function/day	22 days
Pessimistic	0.25 Function/day	45 days

Table 8.2 System Dynamics Estimate

One would say that the above is a very low productivity estimate. Indeed it is. However, we had to take into account that each student was taking at least three other courses and also working on a thesis. Moreover, the students had to spend an excessive amount of time acquiring the necessary programming skills. Therefore, these productivity rates are reasonable. Using the system dynamic model we obtained the results showed in figure 8.5. The medium productivity simulation results confirmed that the designer team estimate was reasonable.

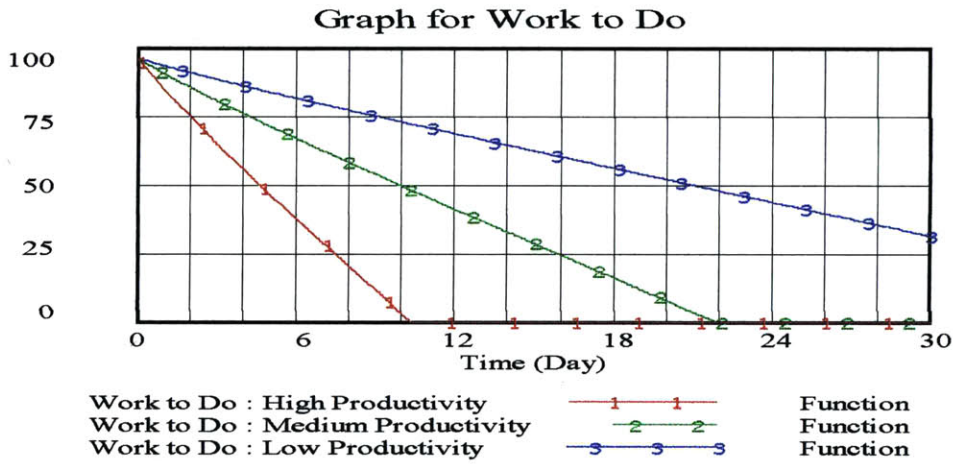

Figure 8.5 Simulation Results

Comparing the estimates given by the model (Figure 8.5) with those given by the team leader, we were able to confirm the duration of the programming activity.

- We spent a significant amount of time learning how to use planning tools (Primavera and Project software), believing that they would help us clarify the plan and control the project. We created several different schedule networks, with different levels of detail. To our surprise, the more detailed the network was, the more confused the students were about the project. Finally we removed from the network all support activities and kept only the activities related to the development of the software. As Goldratt has said in his book (Goldratt, 1997):

  "It's amazing how much the computer software available today simplifies the clerical work. It's also amazing to what extent this sophisticated software doesn't help us to solve the real problem."

  We have to agree with him. The software tools helped define the network of activities and calculate the critical path very fast, but it did not help much in solving many of our project problems, such as controlling the project, for example.
  Concluding, we can say that it is not enough to have a good plan; it is necessary to have effective monitoring and control mechanisms in order to achieve the desired goals. We will cover the monitoring and control aspects in the next section.

8.4 ieCollab Project Monitoring and Control

The monitoring process was based on two different reports: the weekly reports sent by team leaders to the project manager and the weekly meeting, during which the project manager presented to all team members the project manager's weekly report. The team's weekly reports were the basic monitoring tool and the project manager's presentation was the tool used to show all members the control measures taken. The project manager reported weekly on the progress made, the problems faced and the solutions proposed. In order to assure the quality of the development process, the quality assurance team also participated in the project manager presentation. The PDCA model (Figure 8.6) was very effective in ensuring the quality of the development process. Three basic results were achieved by monitoring and controlling: the improvement of

development process, creation of a lessons learned document and the actualization of the project manager plan.

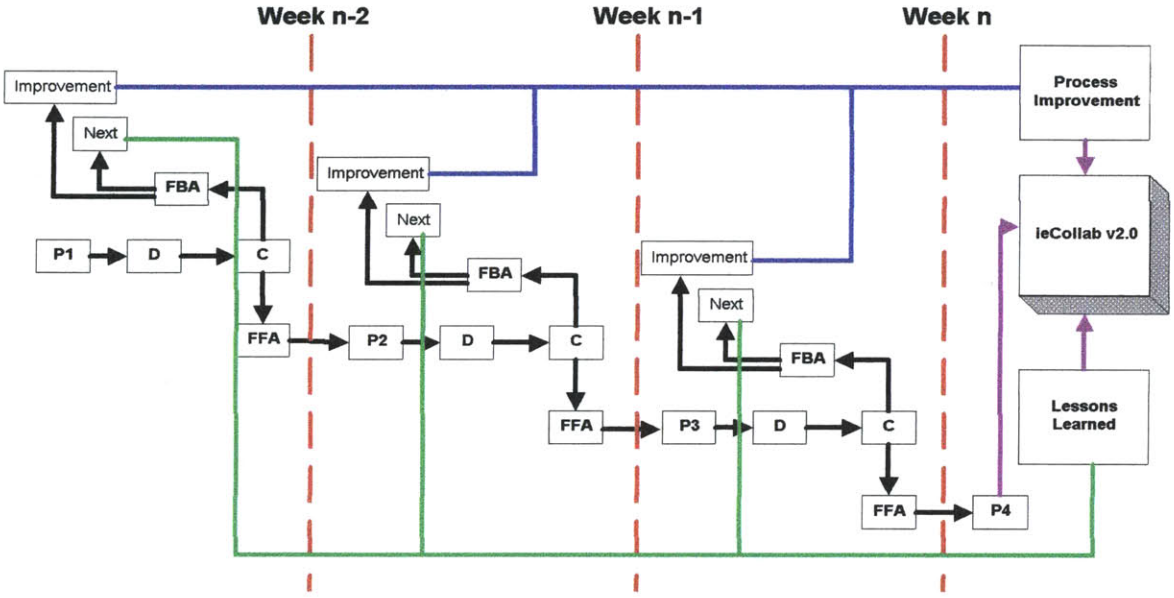

Figure 8.6 The PDCA cycles (Clausing, 1994); (Shiba et al., 1993).

The improvement of the development process was achieved by responding to the feedback from actions taken. These feed-backward actions (FBA) resulted in the improvement of the project development process and in the creation of a lessons learned document. The project management team also had to make many adjustments in the plan along the project. The adjustments in the project management plan were examples of feed-forward actions (FFA). These actions are better understood by means of an example. For example, during weeks 10 to12, the project basically became paralyzed (Figure 8.7). The main reasons for a project paralysis are lack of focus, of direction and of clear priorities (Lewis, 1998). In our case faced had all three problems.

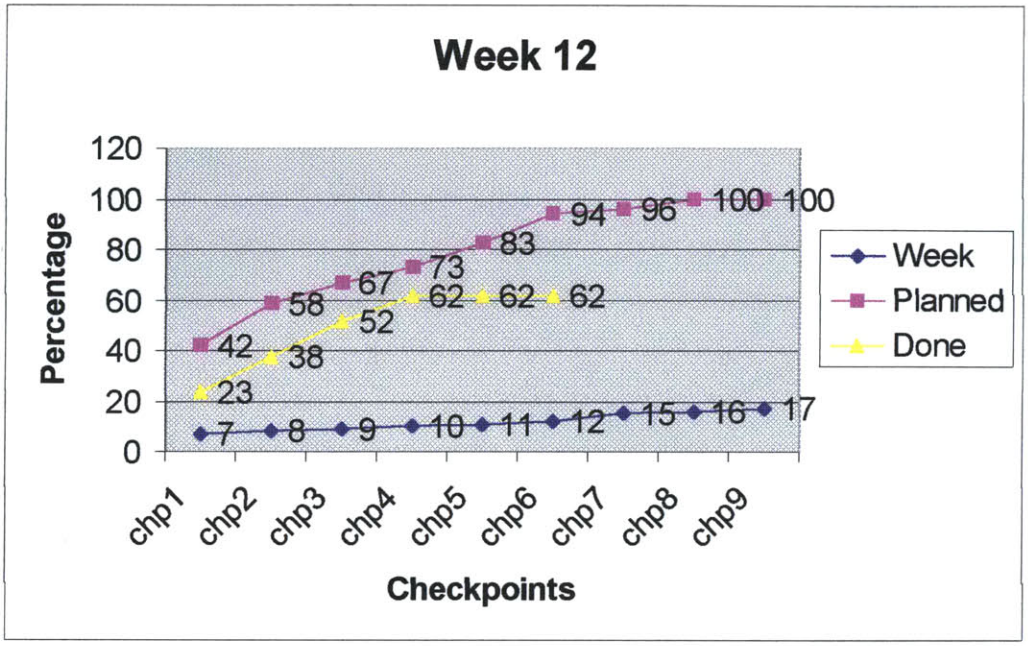

Figure 8.7 Project Paralysis

Several events contributed to the problem. First the head programmer team leader had a problem during week 11 and was not able to define the tasks to his team or send any report to the project management team. As the programming team did not have anyone in charge of coordinating their activities, chaos ensued. The remaining programming team leaders did not take the initiative to meet in order to divide the tasks among themselves. They did not define clear priorities. This caused the second problem, loss of focus. As all students were taking another courses, when they realized the project was having problems, they changed their attention to other courses and activities. Consequently, almost no progress was made during weeks 11 and 12 in terms of programming.

However, good progress was made in terms of control mechanisms. The project management team learned from this problem that the control mechanism could be improved if we created a way to quantify the effort of each programming team (table

8.3). The mechanism that we have created allowed us to measure and inform of the progress of each programming team separately.

	Item	Planned	Done	Problems	Leader
Design MM	2.1	100	100	***	Student A
Design TM	2.2	70	70	***	Student D
Client GUI	3.1.1	50	10	***	Student B
Client Java	3.1.2	50	20	***	Student B
Client Corba	3.1.3	40	30	***	Student B
Serve Corba	3.2.1	30	30	***	Student A
Server Java	3.2.2	10	10	***	Student A
DB SQL/JDBC	3.3.1	40	30	***	Student C
DB Implementation	3.3.2	30	30	***	Student C

Table 8.3 Programming Monitoring

The use of a new control mechanism improved the development process and helped to solve the paralysis problem. The programming team leaders become more active, individual programming assignments were given and the project moved forward.

8.5 ieCollab Project Execution

The ieCollab project was a short duration project. We had to be able to deal with all phases of software development process within few months. In order to achieve success, the project management team tried to employ cycle time reduction techniques without sacrificing quality. In order to improve the development process, we will describe the ieCollab process design using the Center of Quality Management (CQM) guideline (Rasmussen et. al., 1999). This guideline stresses that the main important aspects of process design are leadership, metrics, method, tools and process knowledge.

8.5.1 ieCollab Leadership

In terms of leadership, it became clear to the project management team from the beginning of the project that we needed to rely not only on the team's leader but on the leaders of all other teams. Most of the problems of ieCollab project were due not to a lack of project management measures but to lack of efficient leadership on each of the teams (Smith, 1999). The lack of leadership caused most of the communication problems we encountered. The team leaders were not able to communicate efficiently with their team members. The team leaders did not have a clear idea of what the team members were doing; many times their messages went unanswered. The ieCollab project also suffered from lack of communication between teams, even though of all team members had access to a variety of communication tools (email, Website repository for the project). The tools were available and easy to use; however, they were not used effectively. In order to improve the communication among teams in our weekly meeting, the team members talked about their problems or problems that other teams were causing to their team. The most frequently asked question was whether or not the team leaders of each team had talked each other about their common problems, and the answer most of time was "no". If the team leaders had had more leadership skills, these communications problems would have had overcome easily.

8.5.2 ieCollab Metrics

The utilization of metrics assured not only the measurability of the process but also made the process visible (Rasmussen et. al., 1999). The metrics used on ieCollab, in spite of being simple, were quite efficient[1]. In order to verify the project status we developed two spreadsheets, one with the planned status of each task and other with the actual status of the project (Table 8.4).

[1] The other metrics used were described on Chapter 7.

	Chekpt 01	Chekpt 02	Chekpt 03	Chekpt 04	Chekpt 05	Chekpt 06	Chekpt 07	Chekpt 08	Chekpt 09
Planned									
An. MM	100	100	100	100	100	100	100	100	100
An. TM	50	100	100	100	100	100	100	100	100
Des. MM	50	100	100	100	100	100	100	100	100
Des. TM	50	50	100	100	100	100	100	100	100
Prog	0	0	0	25	50	100	100	100	100
Test	0	0	0	15	45	65	75	100	100
Done									
An. MM	50	100	100	100					
An. TM	50	50	100	100					
Des. MM	25	50	60	100					
Des. TM	15	25	50	100					
Prog	0	0	0	0					
Test	0	0	0	0					

Table 8.4 Measuring Project Status

The actual status of the project was based on feedback given by the team leaders' report. Every week we showed one graph that indicated the project status (see figure 8.7). However, this brought about some unexpected problems. The team leaders realized that this information was visible to all teams and to the faculty. Concerned that they might be faulted for not achieving the expected results, they began to overestimate what they achieved.

8.5.3 ieCollab Method

The software development process was quite good: the project was made visible to all, the weekly meetings and the project management team weekly presentation made clear all our problems and achievements. Each team's weekly report proved to be a very

useful control mechanism; the project management team was kept informed about the other teams' progress, problems, suggestions and amount of time spent on the project.

The weekly report evolved over the project lifetime (Rasmussen et. al., 1999);the report mechanism was not so good the first time it was created. There was a lack of willingness on the part of all members of all teams to make suggestions. As the team members realized that their comments were discussed during our weekly meetings, the quality of their reports improved considerably. Figure 8.8 shows an example of a team weekly report. Basically it gives to the project manager the information about the tasks accomplished, number of hours worked, problems faced and suggestions given.

One negative point to stress here is that the weekly report was perceived to be created by the same person each week. Sometimes the weekly report information reflected his/her personal opinion rather than the teams' opinion.

Another negative aspect of the weekly report is that sometimes the suggestions and comments were not constructive. Some teams misunderstood the nature of the reports and criticized work done by others team in an unexpected way. The project manager team had to filter out these comments in our weekly presentation in order to avoid creating conflict among teams.

WEEKLY CONTROL REPORT

Date:	3/5/2000	ieCOLLAB	

TEAM	Requirement Analysis	WEEK	Week of 2/27/2000	SUBJECT	Progress report
HOURS / WEEK	5				

#	TASKS	STATUS % Accomplished
1	Comment on Design documents	100

Problems:

1. The design was frozen, and would CM have a clear guideline if there is a change request from other teams about the RA documents? For example, is there a change request form? Do the changes have to go through the Change Control Board? How does that work?
2. After the programmers presentation, we think there should be a new schedule for programming after design is finalized and the additional time for testing the code. When is the new programming schedule available?

Suggestions:

1. What is the new project schedule from PM now? How do other teams fit in this new schedule?
2. As suggested in the Testing presentation, they specified that there's not enough communication between RA, Design and their teams. However, RA team has not heard from the Testing team since the beginning of their work. We don't know if they have questions and if they do, we don't know what their questions are.
3. For all the presentations so far, every team has specified that there's a delay due to RA. We want to clarify this issue here. It seems like no one understands the origin of the problem... Therefore, PM team, please clarify our work to the whole ieCollab team ....help us to clarify this to the whole class. Thanks.

APPENDIX 3 – WEEKLY CONTROL REPORT

Figure 8.8 - Weekly Report Example

8.5.4 ieCollab Tools

Every team had the freedom to choose the tools they wanted to use (Table 8.5).

Teams	Tools
Project Manager	• Primavera • MS-Project • Vensim
Analyst	• UML-Rational Rose • Use Case Diagrams
Designer	• UML-Rational Rose • UML-Static and Dynamic Diagrams
Testers	• Test Case
Quality Assurance	• Quality Metrics • Productivity Metrics
Knowledge Manager	• Web repository
Configuration Manager	▪ CVS Software

Table 8.5 ieCollab Tools

The most effective tool was the Web repository (Figure 8.8). All team members could access the documents created by the different teams and add their comments. There were some minor problems with the Website, however. First, sometimes we were unable to put documents there, due to many different connectivity problems. Second, we also had some problems with the use of this Website, many of them related to the communication of what was posted. Although the Website did have a communication tool that allowed sending a message to all subscribers when a new document was posted there, this facility did not always worked properly. Sometimes a team put a document there but this action was not noted by any other teams.

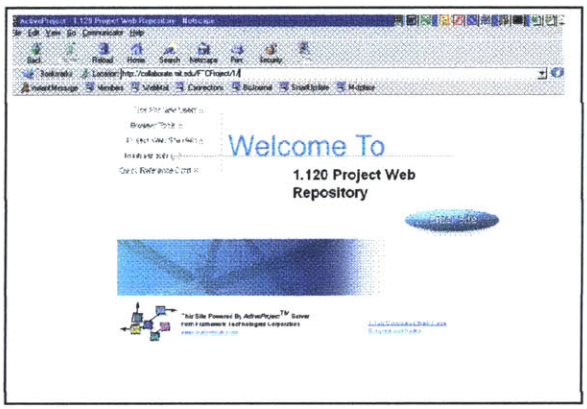

Figure 8.8

In general, however, the Website was very helpful for providing adequate project visibility and for serving as a discussion forum.

8.5.5 ieCollab Virtual Teams Organization

The virtual team structure, which divided leadership of the ieCollab project between CICESE and MIT, worked satisfactorily during the fall term. At the beginning of spring term, however, almost all students from CICESE left the project. The reasons why they left were not clear to us. We guess that it was due to the lack of motivation caused by the difficulties involved in working on the project. The communication with the remaining students began to decrease until it eventually ceased. The CICESE lab had serious Internet connectivity problems at that time; often the email we send to them were lost. The amount of effort required to keep in touch with them began to cause delays in the project. Finally the MIT team decided to finish the project on its own. If at beginning of the project we had assigned someone from the project management team to be responsible for keeping track of the communication between the two teams, this problem would have been reduced. If we had continued to use the real-time Web communication tools and lab that we used during the fall term it would have helped us to keep the contact with Mexico team members.

8.5.6 ieCollab Process Knowledge

The process knowledge can be summarized by means of suggestions that can be useful for future versions of the ieCollab project (Rasmussen et. al., 1999). Table 8.6 gives an overview of do's and don'ts of the ieCollab project in terms of communication, organization, planning and control.

Aspect	Do's	Don'ts
Communication	▪ Have one student from project management team in charge of communication with remotely located students	▪ Don't let communication between teams deteriorate
Organization	▪ Have "link elements" defined at beginning of the project. Make sure to that they are effectively participating in all meetings of both teams.	▪ Don't let the team leaders decide who will be in charge of it. The PM team should choose this "link elements"
Planning	▪ Have all teams participate in the planning process	▪ Don't create the project manager plan only with the project manager team members
Monitoring and control	▪ Make use of audio-visual resources to make the control mechanism visible to all	▪ Don't let the teams send their comments using format different from that provided.

Table 8.6 Do's and Don'ts

8.6 Summary

In this chapter we discussed post-project learning by means of an example, the ieCollab project. We reviewed the lessons learned in all phases of the project, from project preparation to project monitoring and control. The next chapter discusses our final conclusions and suggests areas for future work.

Chapter 9

Conclusions

In this chapter we will discuss a more adaptive approach to managing a high speed, high change software project and apply that approach to ieCollab. After that we will present our final conclusions and give some ideas for areas of future work.

9.1 An improved approach

The project management model described in Chapter 3 is a good conceptual framework. However, based on the lessons learned in the ieCollab project it is not robust to different types of projects. In a recent book (Highsmith, 2000), Highsmith has described a more adaptive software development (ASD) approach. This methodology focuses on managing high speed, high change software projects. The choice of the development approach is based on the mission of the software project. We need to make sure that we understand what our goals are. If we were developing software for equipment that deals with human life (flight control software, medical software and defense system for example), our priorities would be very different from those that we would have if we were developing a Web Browser. In the first case, we would want to avoid any defects, no matter how long it took or how many resources were involved.

In the second case, we could tolerate a certain level of defects. However, an important issue is time to market. The software should be delivered before our competitors. In figure 9.1, we compare the two different software projects; the top left table represents the goals of a medical software application and the right table shows the priorities of a Web Browser. It is clear that the first application is low speed, low change software. The changes are small over the years and the level of defects is kept as small as possible. The Web Browser, on the other hand, interacts with a rapidly changing environment. The features change radically from one version to another. The software must be delivered fast, with as many features as possible in order to be the first on the market. We are not concerned about a low level of defects. The tolerance for defects is much higher. The choice of process model for each type of software (or life cycles) is based on our previous analysis. It makes sense to choose the waterfall[1] model for low speed, low change software. Table 9.1 presents the main characteristics of each process model.

Process Model	**Characteristics**
Waterfall	▪ Requirements are know, risk of change is small ▪ Linear sequential model (Analysis, Design, Code generation, Testing)
Evolutionary	▪ Iterative, increasingly more complex versions ▪ Involves incremental model, spiral model, component assembly model
RAD (rapid application development)	▪ High-speed adaptation of Linear sequential model ▪ Address low-to-moderate level of change environment
Adaptive Software Development	▪ Framework for high-speed, high- change software projects

Table 9.1 -Software process model (Pressman, 1997); (Highsmith, 2000).

[1] We are not going to cover these life cycles in detail. For more detailed explanation we recommend reading the Pressman and Highsmith books (Pressman, 1997); (Highsmith, 2000).

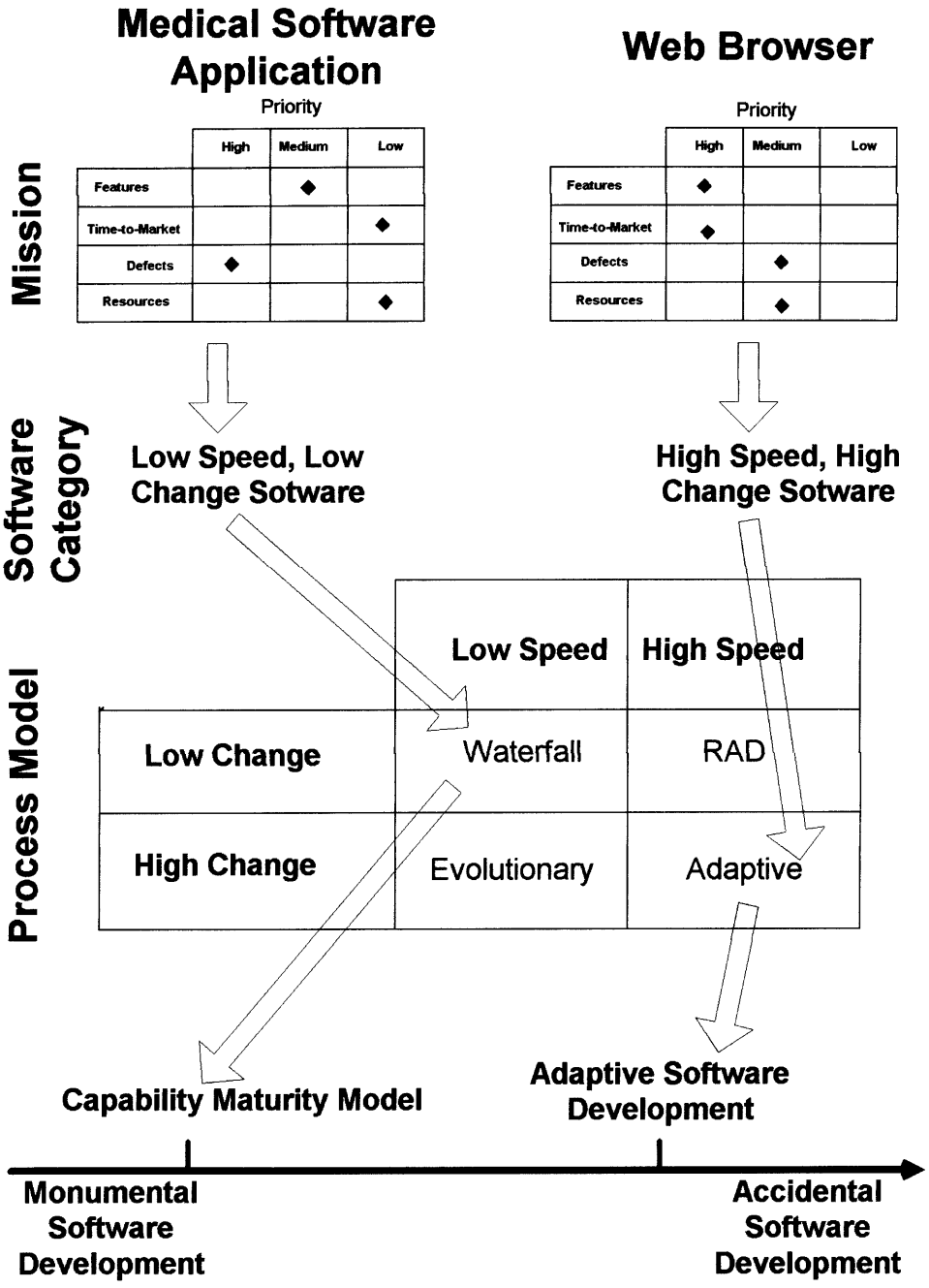

Figure 9.1 Choosing a suitable development approach

Once we have chosen the life cycle, we can choose the more appropriate development process. The development process varies from a totally defined process that Highsmith calls "Monumental Software Development" to a totally chaotic process, the so-called "Accidental Software Development".

The Capability Maturity Model is an example of process development with characteristics of "Monumental Software Development". In 1990, Watts Humphrey (Humphrey, 1990) published the basic framework of capability maturity model (CMM). Since then, CMM has gone through by several refinements, conducted mainly by the Carnegie-Mellon Software Engineering Institute. The CMM is a model that can be used for evaluating the maturity level of software processes of an organization and for improving the software processes as well (Paulk et. al., 1991); (CMU, 1990). The CMM defines five software process maturity levels, each level corresponding to one step of process improvement. The five levels and their characteristics are presented in table 9.2 (Paulk et. al., 1991); (CMU, 1990); (Humphrey, 1990):

Level	**Characteristics**
01 Initial Level	1. Absence of stable environment for developing software projects 2. Software process is constantly changed or modified as the work progresses 3. Schedule, functionality and product quality are unpredictable 4. Few stable software processes in evidence
02 Repeatable level	1. Policies for managing software process established 2. Planning and managing new projects is based on experience with similar projects 3. There are basic software management controls 4. There is a tracking mechanism for costs, schedules and functionality 5. Software project's standards are defined and followed
03 Defined level	1. There is a standard documented process for developing and maintaining software 2. There is a software engineering process group responsible for organizing software process activities

	3. Management activities and software engineering activities are stable and repeatable
04 Managed level	1. There are quantitative quality goals for software products and processes defined 2. There is database used to collect and analyze the data available from the software project
05 Optimizing level	1. The organization is focused on continuous process improvement 2. There is a defined way to analyze defects and to determine their causes

Table 9.2- CMM level characteristics (Paulk et. al., 1991); (CMU, 1990); (Humphrey, 1990)

The CMM emphasis is on process improvement. It assumes that the process is visible, repeatable, measurable and that the software works in a stable environment. The next software version will follow the same steps as the previous one. However, as the development become more and more rigorous, the software applicability becomes more and more narrow (Highsmith, 2000).

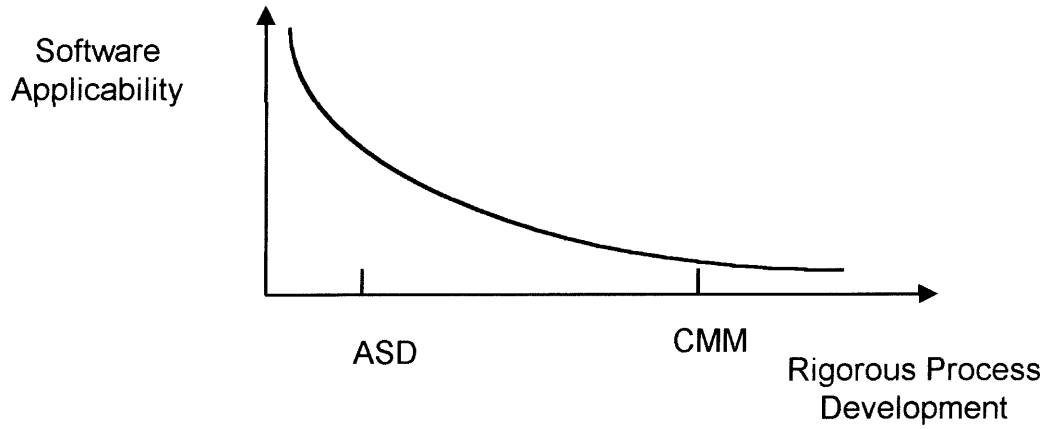

Figure 9.3 Process Development x Software Applicability

The CMM methodology is very useful for creation of software that deals with human life. We want that this kind of software to follow a very rigid structure, in order to avoid any defects. However, this methodology seems not to be appropriate for software

that does not require that degree of safety and that works in a rapidly changing environment like the Web. In this situation, adaptive software development seems to be more appropriate. Figure 9.4 shows the basic ideas of adaptive software development.

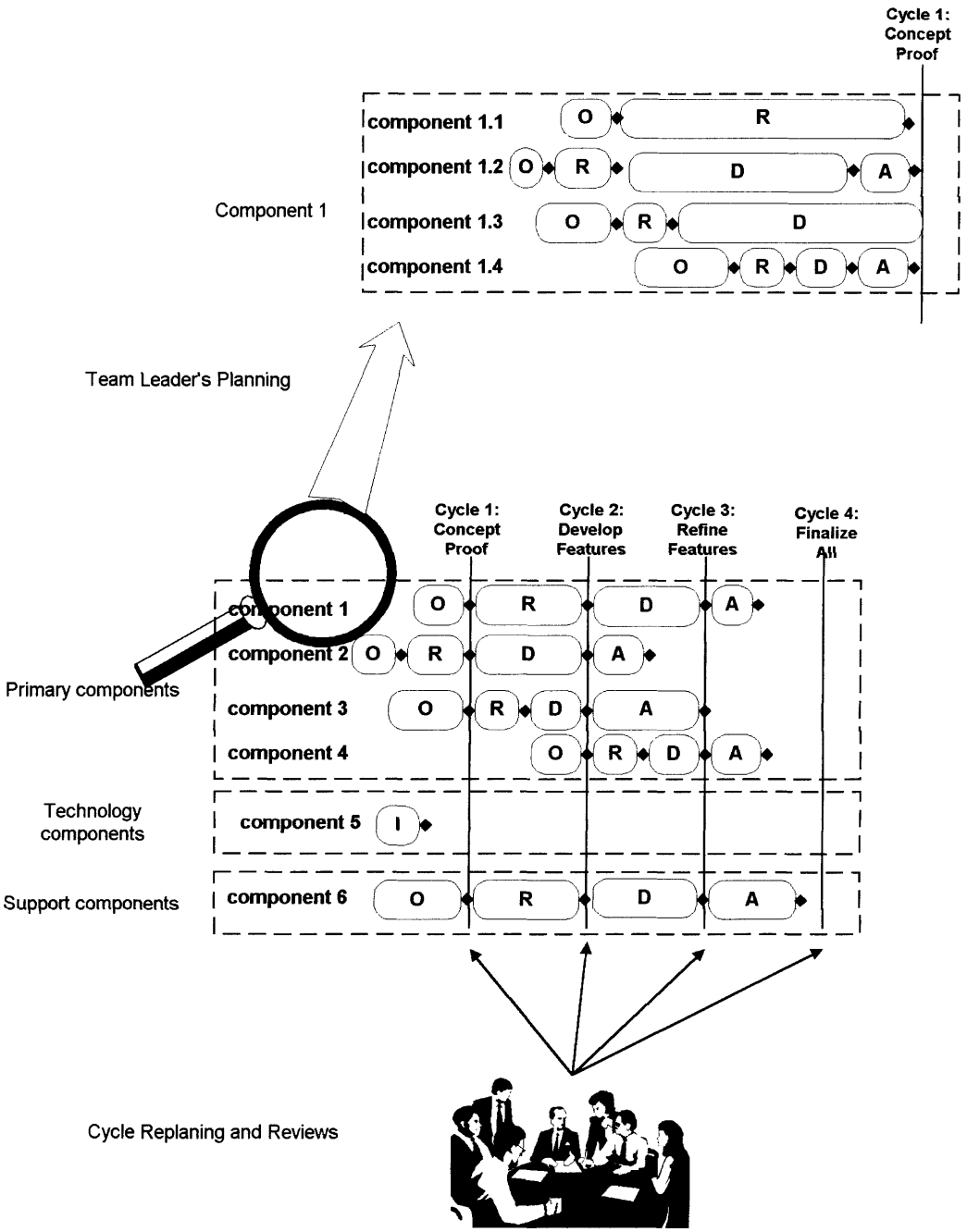

Figure 9.4- The ASD Methodology

The most interesting concepts of this methodology are the development of adaptive cycles and the choice of results-oriented rather than task-oriented project management. Figure 9.4 shows that during project preparation, the project is divided into components. The components are divided into primary components, technology components and support components. Highsmith defines the component as follows (Table 9.3):

Component Type	**Characteristics**	**Examples**
Primary	Deliver functionality to customer	▪ Class and its methods ▪ Use Case ▪ Sequence Diagram
Technology	Components on which the primary components are built.	▪ Operational System ▪ Compiler ▪ Installation of a database system
Support	Support documents	▪ Technical Manual ▪ User's Guide ▪ Project Plans

Table 9.3 Components Types (Highsmith, 2000)

For example, in figure 9.4 we could have the following components:

Component	**Type**
01-Graphical Interfaces	Primary
02-Analysis	Primary
03-Design	Primary
04-Test	Primary
05-Installation of C++	Technology
06- Technical Manual	Support

Table 9.5- Project components

The circles represent the state of each component. We are using the following convention:

State	Convention
Outline	O
Review	R
Detail	D
Approval	A
Installation	I

Table 9.6- Components states

The initial plan also defines the scope of each cycle. For example, at the end of cycle 1, the teams will have a clear idea of the scope of the project and its feasibility. The cycle is called "Concept Proof". Looking at figure 9.4, it shows that the teams will have the graphical interface in a "outline" stage. This means they will have a very simple prototype that gives an idea of what the final product will look like. The teams will also have an Analysis document in its review phase and a Design Document in its outline phase. The C++ program has already been installed and the outline of the technical manual have been defined. The plan is not detailed. The teams have the freedom to decide what they have to do in order achieve the goals of this cycle. The top part of figure 9.4 shows a possible planning defined by the programming team leader. He/She could use the same methodology to define the functions to be created. The re-planning and review process is done at the end of each cycle.

In the following sections we will cover what we would have done differently in terms of project preparation, project planning and project execution if we had applied the ASD methodology.

Project Preparation

In this phase we did not do any feasibility study; we defined the projects' scope, but we did not spend time analyzing if it was feasible to achieve or not. Perhaps because we were overconfident about our skills, or perhaps because we did not know how complex the project we were about to face would be. If we were to do it again, we would do a feasibility study; this way we could define a more realistic project scope.

At this phase team members should learn about the basic concepts of client-server development. If all team members had the basic ideas of client-server architecture (transactions models, three tier client/server model, network protocols for example) it would have avoided many misunderstandings that we had during all project.

Another aspect to cover here is that we did not have a shared vision of the project's scope. The scope was defined by the Business Management team and the Marketing Management team and enforced by the Project Management team. It would have been more effective to have the participation of all team members.

Another aspect that we did not cover in ieCollab was how to define project success. The traditional approach to defining success is that the project meet one or all of the following criteria: it performs the requirements; it is delivered on time, on budget and with quality; and it provides the team with professional satisfaction and the opportunity to learn (Highsmith, 2000).

The ieCollab project was an educational project and the success definition is quite different. Our main goal was to emulate a real project, and to make all students familiar with the software development process and learn about the roles they were performing. In terms of this project success criteria, we could say that ieCollab was successful. However, the success criterion was not stated at the project preparation phase. Because of this many students felt frustrated at the end, because the software we developed was not able to meet the requirements we have set for it.

Project Planning

It is unrealistic to believe that it is possible to plan all the activities involved in a high speed, high change software projects. The environment and the competition change so fast, that not all activities can be determined. The teams have an idea of the core activities but the details are unknown. It is better to have a plan that establishes the basic components of the project than to focus in the functions of each one. Our planning approach would have been better if we had done a better job of keeping the documents under better configuration control. In the ieCollab project, we had serious problems tracking the state of each component involved in the project. For example, at one milestone, the Designer team was supposed to produce the design documents about transaction management functions. It was released as version "Design Document, version 1.0". Some days later the group released a new version, version 1.1 with substantial changes. And this process continued until the end of project. At the end of project we had Analysis documents version 1.5, design documents version 1.3, project management plan version 2.0 and so on. It was very troublesome to determine which documents were final, which ones had been approved, which ones were under review and so on. If we had defined only four states, and made sure to have the components delivered at each milestone, the management would have been much simpler.

Highsmith divides project components into three categories (primary, technology and support) and four states (outline, detail, reviewed and approved). Short cycles with clear objectives and clear definition of the state of the components are better than larger cycles with very detailed documents. In the ieCollab project we spent a lot of time in detailing the analysis and design documents. While the Design and Analysis teams were spending a lot of energy and time on the project, many team members were doing little. Figure 9.5 shows how inefficient the resource allocation of ieCollab was. Each team was focused on optimizing their job rather than on optimizing the final product. There was a discontinuity in the work as well. For example, students who belonged to the business manager team worked effectively during the first two weeks of the project. In their second role as programmers, however, they were not productive for many weeks. This was caused by a snowball effect: the more time analysts spent creating documents, the

more delay the designers faced. While the designers were struggling to create very precise diagrams, the programmers were doing almost nothing. Only at the end of the project did we have all students working.

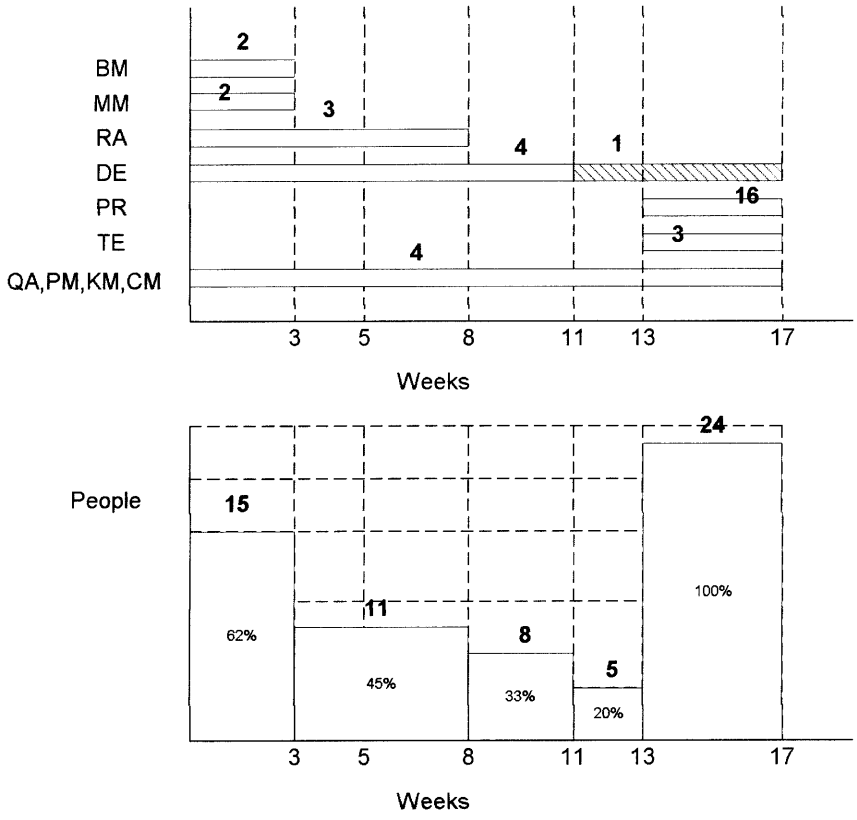

Figure 9.5 -The ieCollab Resource Allocation

It would be much more interesting to do a small part of the analysis, design, coding and testing at each cycle. At the end of ieCollab project we had completed all components. However, it was not possible to integrate them into a final product. This problem was in partly caused by our task-management approach.

If we were going to do the planning again, we would create component oriented planning with clearly defined states and cycle milestones (Table 9.7).

	Cycle 1	Cycle 2	Cycle 3	Cycle 4
Primary Components				
01-Client GUI	Review	Detail	Approval	
02-Client Java Classes	Review	Detail	Approval	
03-Client Corba Interface	Outline	Review	Approval	
04-Server Corba Interface	Outline	Review	Detail	Approval
05-Server Java Classes	Outline	Review	Detail	Approval
06-Interface JDBC/SQL	Outline	Review	Approval	
07-Database Implementation	Review	Detail	Approval	
Technology Components				
08-Web Repository	Approval			
09-Configuration Management Repository	Detail	Approval		
10-Oracle Database	Review	Approval		
11-Communication Tools	Approval			
12-Rational Rose	Approval			
Support Components				
13- Project Management Plan	Review	Approval		
14-Analysis Document	Review	Detail	Approval	
15-Design Documents	Outline	Review	Detail	Approval
16- Testing Documents	Outline	Review	Detail	Approval
17-Users Guide	Outline	Review	Detail	Approval
18-Technical Guide	Outline	Review	Detail	Approval

Table 9.7- ieCollab Components

At the end of each project cycle, the project team would have a prototype. The advantage of this approach is that the prototype could be shown to the client. This way we could be sure that the software would meet the clients needs. Another advantage is that this approach forces the integration of diverse components at an early stage of development.

Project execution

The use of ASD methodology makes the monitoring and control simpler. The project manager defines the components and the deadlines: the teams define how to create the components. This way the project manager can have a clear overview of all project components[2] and can track the project based on results rather than tasks (Figure 9.6).

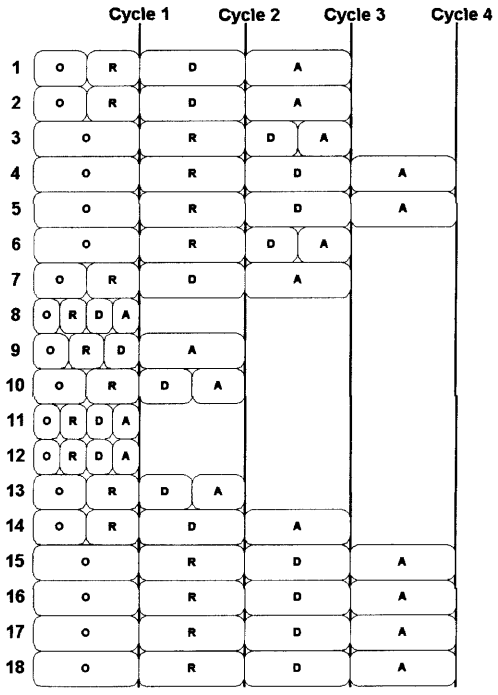

Figure 9.6- ieCollab component state diagram

[2] In figure 9.6, O means "outline", R means "review", D means "detail" and A means "approved"

The project manager does not need to define the components in detail. That is done by the teams, in the way that they find most effective. This improves creativity and gives freedom to the team members. The role of the project manager is to provide direction, to define the boundaries and to define the deliverables. The project manager will not be responsible for determining exactly how it will be done, which classes are part of each component or how they work together. The project manager will check if the product has the expected characteristics at the end of each cycle.

9.2 Conclusions

This thesis covered many of the principles, methods and tools used in the management of distributed collaborative project. In the beginning of our work we reviewed the main issues, basic ideas and definitions involved in project management. The initial part of the thesis was written during the fall term of 1999, while taking a project management course at MIT. At this time I did not have experience as a project manager, and was trying to understand the main aspects involved in all steps of project management. The course taken in project management during the fall term was very useful. However, the real learning came with the work in the ieCollab project as a member of the project management team and writing about this experience.

As the thesis evolved, I tried to reduce the level of abstraction by giving as many examples as possible, showing how we applied the methods and tools in the ieCollab Project. The ieCollab project helped us understand the dynamic nature of planning, and the need to have an efficient monitoring and control mechanism. Working on the ieCollab project and at the same time writing a thesis was a very rich experience. I was studying the methods and tools and applying them at same time. Some tools proved to be appropriate to the project; others were not very useful. The experience acquired as member of the project management team of the ieCollab project was fundamental in understanding the difference between the theory and practice of project management.

The two main lessons learned was that while mastering the tools is helpful, this does not guarantee the success of the project. The other lesson is that project planning

and execution differs with the nature of the project. The tools used to manage repetitive software projects in a stable environment are not appropriate for managing high-speed, high-change software projects in a fast changing environment. Managing software projects is a challenge; managing a virtual team based project like the ieCollab project was much more challenging.

9.3 Areas of Future Work

In this thesis we have discussed project management tools that can be used internally by an organization. We did not discuss how to allow collaborative project planning and scheduling among different companies. Software tools developed by companies such as I2, Manugistics, SAP, Oracle promise to allow to integrated project management, production planning and scheduling. In the future, it will be interesting to explore how adequate these tools are in helping to manage collaborative distributed software projects.

References

[Amaral et. al., 1999] Amaral, J.A.A., Limansky I. and Abbot E.(1999).*The ieCollab Project Management Plan*, Retrieved May 25, 2000 from the World Wide Web: http://www.collaborate.mit.edu/FTP/P1

[Benson-Armer, 1997] Benson-Armer, R. A. and Hsieh T.Y. (1997). Teamwork across time and space. *The McKinsey Quarterly*, (4), 19-27.

[Clausing, 1994] Clausing, D.(1994). *Total Quality Development: a step-by-step guide to world-class concurrent engineering.* New York, USA: ASME Press.

[Connolly et. al.,2000] Connolly T.and Begg C.(2000). *Database Solutions: a step-by-step guide to building databases*. London, UK: Addison-Wesley.

[CMU, 1990] Carnegie-Mellon Software Engineering Institute (1990). *Capability Maturity Model (SW-CMM) for Software*. Retrieved April 21,2000 from the World Wide Web: http://www.sei.cmu.edu:80/cmm/cmm.html

[Forrester, 1961] Forrester, Jay.W. (1961). *Industrial Dynamics.* Portland, USA: Productivity Press.

[Forrester, 1971] Forrester, Jay W.(1971). *Counterintuitive Behavior of Social Systems,* System Dynamic Lectures Notes, Sloan School of Management, Massachusetts Institute of Technology, Cambridge, USA.

[Forrester , 1999] Forrester, Jay W.(1999). *Introduction to System Dynamics*, System Dynamic Lectures Notes, Sloan School of Management, Massachusetts Institute of Technology, Cambridge, USA.

[Goldratt, 1997] Goldratt, E.M. (1997). *Critical Chain*. Great Barrington, USA: The north river press.

[Grantham, 2000] Grantham,C.E. (2000). *The future work.* Retrieved May 25, 2000 from the World Wide Web http://www.isdw.com

[Highsmith, 2000] Highsmith III, J. A. (2000). *Adaptive Software Development: a collaborative approach to managing complex systems*, New York, USA: Dorset House Publishing Company.

[Hines,1999] Hines, J. (1999). *The Standard Method,* Application of System Dynamics Lectures Notes, Sloan School of Management, Massachusetts Institute of Technology, Cambridge, USA.

[Horvath et. al.,1999] Horvath,L. and Tobin T.J.(1999). *Twenty-first century teamwork: defining competencies for virtual teams*, Paper presented at the 1999 Academy of Management Shared Interest Track Organization Development and Change, USA.

[Humphrey,1990] Humphrey W.S. (1990). *Managing the Software Process*, Berkeley, USA: Addison-Wesley

[Krezner, 1989] Krezner,H.(1989). *Project Management a system approach to planning, scheduling and controlling*. New York, USA: Van Nostrand Reinhold.

[Lewis, 1998] Lewis, J. (1998,September/October). Participative leadership or project paralysis. *Women in Business*, 50(5), 36-39.

[Limansky et al. 1999] Limansky I. and all students of the MIT Master of Engineering Information Technology Class of 2000 (1999), *The IeCollab project.* Retrieved May 25, 2000 from the World Wide Web: http://www.collaborate.mit.edu/FTP/P1

[Lyneis, 1999] Lyneis, J. (1999). *Dynamics of Project Performance.* System and Project Management Course, Lectures Notes, Civil and Environmental Engineering Department, Massachusetts Institute of Technology, Cambridge, USA.

[McConnell, 1996] McConnel,S.C.(1996). *Rapid Development: taming wild software schedules*. New York, USA: Microsoft Press.

[Meredith et. al. 1989] Meredith, J. and Mantel S. (1989). *Project Management: A managerial approach*. New York, USA: Jon Wiley & Sons.

[Paulk et. al., 1991] Paulk, M.C., Weber C.V., Garcia S.M., Chrissis M.B and Bush M.(1991). *Key Practices of the Capability Maturity Model*^{SM}, *Version 1.1.* Retrieved May 25,2000 from the World Wide Web: http://www.sei.cmu.edu:80/cmm/cmm.html

[Peña-Mora et. al., 1999] Peña-Mora, F., Lyneis J. and Warmkessel J.(1999). System and Project Management Course, Lectures Notes, Civil and Environmental Engineering Department, Massachusetts Institute of Technology, Cambridge, USA.

[Pressman, 1997] Pressman, R.S. (1997). *Software Engineering: A practitioner's approach*. New York, USA: McGraw-Hill.

[Rasmussen et. al,1999] Rasmussen N., Walden D.(1999). *Observations from 1997-1998 CQM Study-Group on Cycle Time Reduction.* Center for Quality of Management Journal on line. Retrieved May 25,2000 from the World Wide Web: http://www.cqmextra.cqm.org/cqmjournal.nsf/reprints/rp1000

[Roman, 1986] Roman D.D. (1986). *Managing projects: A system approach*. New York, USA: Elsevier

[Rosenau et. al.,1993] Rosenau D.M. and Moran J.J. (1993). *Managing the Development of New Products: Achieving Speed and Quality Simultaneously through Multifunctional Teamwork*. New York: USA, Van Nostrand Reinhold.

[Rosenau, 1998] Rosenau D. M. (1998). *Successful Project Management: a Step-by-step Approach with Practical Examples*. New York, USA: John Wiley & Sons

[Senge, 1990] Senge, Peter M.(Fall,1990). The Learder's New Work: Building Learning Organizations. *Sloan Management Review*, 32(1), 7-23.

[Shiba et. al. 1993] Shiba S., Grahan A. and Walden D. (1993). *A New American TQM: Four Practical Revolutions in Management.* Portland, USA: Productivity Press.

[Shtub et. al., 1994] Shtub A., Bard J.F. and Globerson S. (1994). *Project Management Engineering, Technology and Implementation.* New Jersey, USA: Prentice Hall.

[Smith et. al., 1998] Smith, P.G and Reinertsen D.G. (1998). *Developing products in half of the time: new rules, new tools.* New York, USA: John Wiley and Sons.

[Smith, 1999] Smith G. (1999, August). Project leadership: Why project management alone doesn't work, *Hospital Materiel Management Quarterly*, 21(1), 88-92.

[Sterman, 1992] Sterman, John D. (1992). *System Dynamics Modeling for Project Management.* System Dynamic Group, Sloan School of Management, Massachusetts Institute of Technology, Cambridge, USA.

[Sterman, 1998] Sterman, John D. (1998). *Business Dynamics: System thinking and modeling for a complex word*, Partial Draft, Version 1.2, System Dynamic Group, Sloan School of Management, Massachusetts Institute of Technology, Cambridge, USA.

[Thamhain, 1992] Thamhain, H.J. (1992). *Engineering Management: Managing effectively in technology-based organizations.* New York, USA: Wiley Series in Engineering and Technology Management.

[Thomsett, 1990] Thomsett C.M. (1990). *The Little Black Book of Project Management.* New York, USA: Amacom

[Townsend et. al., 1998] Townsend,A.M, DeMarie,S.M. and Hendrickson, A.R.(1998). *Virtual Teams: Technology and the workplace of the future, Academy of Management Executive*, 12,(3), 17-29.